Enabling Quality-of-Service Applications in Sensor Networks

A Thesis
Presented to
The Academic Faculty

by

Weilian Su

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in Electrical and Computer Engineering

School of Electrical and Computer Engineering
Georgia Institute of Technology
April 2004

Enabling Quality-of-Service Applications in Sensor Networks

Approved by:

Professor Ian F. Akyildiz, Chairman

Professor Ye (Geofferey) Li

Professor Yorai Y. Wardi

Professor Raghupathy Sivakumar

Professor Mostafa H. Ammar

Date Approved: 2 April 2004

*To my **Master** who constantly*

encourages me to improve my inner self and value virtue regardless

of the hardships and obstacles, to my mom, dad, and brothers for their

unquestionable love and support, and to my friends

for their life-long companionship.

Acknowledgements

I want to thank Dr. Ian F. Akyildiz for his encouragement and support for the past five years. He has been a great teacher as well as a friend. I know he uses a lot of energy trying to improve and teach me the skills that are essential for a successful career. From the bottom of my heart, I sincerely thank him for his patience and belief in my talents.

Also, I want to thank all my friends for their support and encouragement. I want to especially thank my friends in NYC and Marina K. Thottan for their mental and emotional support.

Furthermore, I want to thank my Ph.d. dissertation committee, Dr. Ye (Geofferey) Li, Dr. Raghupathy Sivakumar, Dr. Ramesh Jain, Dr. Yorai Y. Wardi, and Dr. Mostafa H. Ammar, for their guidance and invaluable comments. I would like to thank Dr. Geofferey Li and Dr. Raghupathy Sivakumar for their insights into the academic world.

In addition, I want to thank the members of the Broadband and Wireless Networking Laboratory for their support and friendships. Especially, I want to give my thanks to Xudong Wang, Jaudelice Cavalcante de Oliveira, Jiang (Linda) Xie, Tricha Anjali, and Ozgur Baris Akan for their wisdom and helping hands. Also, I want to thank Cordai Farrar and Kesha Jackson for their help.

Lastly, I want to thank my brothers and parents for their support and encouragement. Also, I would like to thank all the practitioners of Falun Dafa for their support and enlightenment in becoming a better person. I want to give my salute to the practitioners who were along my side during hunger strike and rallies under rain, snow, hot sun, chill wind, and thunder – voicing for *human rights*.

Table of Contents

List of Tables

List of Figures

Summary

Recent advances in Micro Electro-Mechanical Systems technology, wireless communications, and digital electronics have enabled the development of low-cost, low-power, multifunctional sensor nodes that are small in size and communicate untethered in short distances. These tiny sensor nodes, which consist of sensing, data processing, and communicating components, leverage the idea of sensor networks based on collaborative effort of a large number of nodes. A wide range of applications utilizing low-end sensor nodes to collaborative work together is envisioned for sensor networks. Some of the application areas are health, military, and security. For example, sensor networks can be used to detect foreign chemical agents in the air and the water. They can help to identify the type, concentration, and location of pollutants. In essence, sensor networks will provide the end user with intelligence and a better understanding of the environment. Realization of these and other sensor network applications require certain fundamental protocols and schemes. The objective of this thesis is to provide some of the basic building blocks that are necessary for sensor networks. These basic blocks are in the areas of *routing*, *time synchronization*, and *localization*. The routing protocol allows different types of traffics to be delivered and fused during delivery to lower the amount of information exchange. The time synchronization protocol enables the sensor nodes to maintain a similar time while the localization technique provides a way to find the sensor nodes in the sensor field. The routing, time synchronization, and localization schemes may be used to provide Quality-of-Service when data is gathered from the sensor networks.

Chapter 1

Introduction

Sensor networks have attracted many attentions because of their potential applications [2]. At the same time, the applications have driven many developments in research. This thesis provides three major research contributions that enable Quality-of-Service (QoS) applications in sensor networks.

1.1 Sensor Networks

Recent advances in Micro Electro-Mechanical Systems technology, wireless communications, and digital electronics have enabled the development of low-cost, low-power, multifunctional sensor nodes that are small in size and communicate untethered in short distances. These tiny sensor nodes, which consist of sensing, data processing, and communicating components, leverage the idea of sensor networks based on collaborative effort of a large number of nodes. Sensor networks represent a significant improvement over traditional sensors, which are deployed in the following two ways [37]:

- Sensors can be positioned far from the actual *phenomenon*, i.e., something known by sense perception. In this approach, large sensors that use some complex techniques to distinguish the targets from environmental noise are required.
- Several sensors that perform only sensing can be deployed. The positions of the sensors and communications topology are carefully engineered. They transmit time series of the sensed phenomenon to the central nodes where computations are performed and data are fused.

A sensor network is composed of a large number of sensor nodes, which are densely deployed either inside the phenomenon or very close to it. The position of sensor

nodes need not be engineered or pre-determined. This allows random deployment in inaccessible terrains or disaster relief operations. On the other hand, this also means that sensor network protocols and algorithms must possess self-organizing capabilities. Another unique feature of sensor networks is the cooperative effort of sensor nodes. Sensor nodes are fitted with an on-board processor. Instead of sending the raw data to the nodes responsible for the fusion, sensor nodes use their processing abilities to locally carry out simple computations and transmit only the required and partially processed data.

Realization of these features requires wireless ad hoc networking techniques. Although many protocols and algorithms have been proposed for traditional wireless ad hoc networks, they are not well suited for the unique features and application requirements of sensor networks. To illustrate this point, the differences between sensor networks and ad-hoc networks [56] are outlined below:

- The number of sensor nodes in a sensor network can be several orders of magnitude higher than the nodes in an ad hoc network.
- Sensor nodes are densely deployed.
- Sensor nodes are prone to failures.
- The topology of a sensor network changes very frequently.
- Sensor nodes mainly use broadcast communication paradigm whereas most ad hoc networks are based on point-to-point communications.
- Sensor nodes are limited in power, computational capacities, and memory.
- Sensor nodes may not have global *identification* (ID) because of the large amount of overhead and large number of sensors.

The above differences need to be addressed in order to realize the different types of applications. In Section 1.1.1, the potential applications of sensor networks are provided, and the sensor networks' communication architecture is described in Section 1.1.2.

1.1.1 Sensor Networks Applications

Sensor networks may consist of many different types of sensors such as seismic, low sampling rate magnetic, thermal, visual, infrared, acoustic and radar, which are able to monitor a wide variety of ambient conditions that include the following [24]: temperature, humidity, vehicular movement, lightning condition, pressure, soil makeup, noise levels, the presence or absence of certain kinds of objects, mechanical stress levels on attached objects, and the current characteristics such as speed, direction, and size of an object.

Sensor nodes can be used for continuous sensing, event detection, event identification, location sensing, and local control of actuators. The concept of microsensing and wireless connection of these nodes promise many new application areas such as military, environment, health, and home. It is also possible to expand this classification with more categories such as space exploration, chemical processing and disaster relief.

1.1.1.1 Military Applications

Wireless sensor networks can be an integral part of military *command, control, communications, computing, intelligence, surveillance, reconnaissance and targeting* (C4ISRT) systems. The rapid deployment, self organization and fault tolerance characteristics of sensor networks make them a very promising sensing technique for military C4ISRT. Since sensor networks are based on the dense deployment of disposable and low cost sensor nodes, destruction of some nodes by hostile actions does not affect a military operation as much as the destruction of a traditional sensor, which makes sensor networks concept a better approach for battlefields. Some of the military applications of sensor networks are monitoring friendly forces, equipment and ammunition; battlefield surveillance; reconnaissance of opposing forces and terrain;

targeting; battle damage assessment; and nuclear, biological and chemical attack detection and reconnaissance.

1.1.1.2 Environmental Applications

Some of the environmental applications are forest fire detection, biocomplexity mapping of the environment, and flood detection, which are explained as follows:

Forest Fire Detection - Since sensor nodes may be strategically, randomly, and densely deployed in a forest, sensor nodes can relay the exact origin of the fire to the end users before the fire is spread uncontrollable. Millions of sensor nodes can be deployed and integrated using radio frequencies/optical systems. Also, they may be equipped with effective power scavenging methods [12], such as solar cells, because the sensors may be left unattended for months and even years. The sensor nodes will collaborate with each other to perform distributed sensing and overcome obstacles, such as trees and rocks, that block wired sensors' line-of-sight.

Biocomplexity Mapping of the Environment [14] - A biocomplexity mapping of the environment requires sophisticated approaches to integrate information across temporal and spatial scales [73, 27]. The advances of technology in the remote sensing and automated data collection have enabled higher spatial, spectral, and temporal resolution at a geometrically declining cost per unit area [15]. Along with these advances, the sensor nodes also have the ability to connect with the Internet, which allows remote users to control, monitor and observe the biocomplexity of the environment.

Although satellite and airborne sensors are useful in observing large biodiversity, e.g., spatial complexity of dominant plant species, they are not fine grain enough to observe small size biodiversity, which makes up most of the biodiversity in an ecosystem [41]. As a result, there is a need for ground level deployment of wireless sensor nodes to observe the biocomplexity [29, 30]. One example of biocomplexity

mapping of the environment is done at the James Reserve in Southern California [14]. Three monitoring grids with each having 25 to 100 sensor nodes will be implemented for fixed view multimedia and environmental sensor data loggers.

Flood Detection [8] - An example of a flood detection is the ALERT system [74] deployed in the U.S. Several types of sensors deployed in the ALERT system are rainfall, water level and weather sensors. These sensors supply information to the centralized database system in a predefined way. Research projects, such as the COUGAR Device Database Project at Cornell University [8] and the DataSpace project at Rutgers [36], are investigating distributed approaches in interacting with sensor nodes in the sensor field to provide snapshot and long-running queries.

1.1.1.3 Health Applications

Sensor nodes can also be used to improve the quality of the health system. Some of their applications are telemonitoring of human physiological data, tracking and monitoring doctors and patients inside a hospital, and drug administration in hospitals.

Telemonitoring of Human Physiological Data - The physiological data collected by the sensor networks can be stored for a long period of time [39], and can be used for medical exploration [53]. The installed sensor networks can also monitor and detect elderly people's behavior, e.g., a fall [11, 16]. These small sensor nodes allow the subject a greater freedom of movement and allow doctors to identify pre-defined symptoms earlier [50]. Also, they facilitate a higher quality of life for the subjects compared to the treatment centers [5]. A "Health Smart Home" is designed in the Faculty of Medicine in Grenoble-France to validate the feasibility of such system [51].

Tracking and Monitoring Doctors and Patients Inside a Hospital - Each patient has small and light weight sensor nodes attached to them. Each sensor node has its specific task. For example, one sensor node may be detecting the heart rate while another is detecting the blood pressure. Doctors may also carry a sensor node,

which allows other doctors to locate them within the hospital.

Drug Administration in Hospitals - If sensor nodes can be attached to medications, the chance of getting and prescribing the wrong medication to patients can be minimized. Because, patients will have sensor nodes that identify their allergies and required medications. Computerized systems as described in [63] have shown that they can help minimize adverse drug events.

1.1.1.4 Home Applications

Besides military, environmental, and health applications, sensor networks can improve our way of life in many home applications such as home automation and smart environment.

Home Automation - As technology advances, smart sensor nodes and actuators can be buried in appliances, such as vacuum cleaners, microwave ovens, refrigerators, and VCRs [58]. These sensor nodes inside the domestic devices can interact with each other and with the external network via the Internet or Satellite. They allow end users to manage home devices locally and remotely more easily.

Smart Environment - The design of smart environment can have two different perspectives, i.e., human-centered and technology-centered [1]. For human-centered, a smart environment has to adapt to the needs of the end users in terms of input/output capabilities. For technology-centered, new hardware technologies, networking solutions, and middleware services have to be developed. A scenario of how sensor nodes can be used to create a smart environment is described in [34]. The sensor nodes can be embedded into furniture and appliances, and they can communicate with each other and the room server. The room server can also communicate with other room servers to learn about the services they offered, e.g., printing, scanning, and faxing. These room servers and sensor nodes can be integrated with existing embedded devices to become self-organizing, self-regulated, and adaptive systems based on control

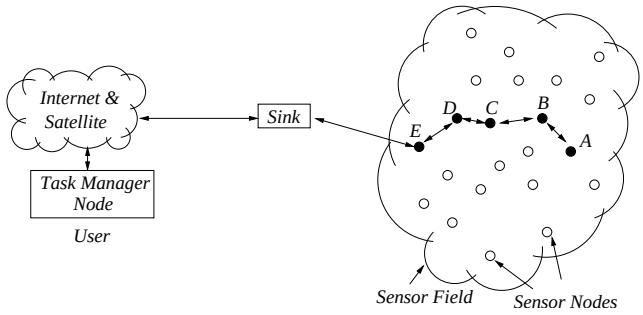

Figure 1: Sensor nodes scattered in a sensor field.

theory models as described in [34]. Another example of smart environment is the "Residential Laboratory" at Georgia Institute of Technology [23]. The computing and sensing in this environment has to be reliable, persistent, and transparent.

1.1.2 Sensor Networks Communication Architecture

The sensor nodes are usually scattered in a *sensor field* as shown in Figure 1. Each of these scattered sensor nodes has the capabilities to collect data and route data back to the *sink* and the end users. Data are routed back to the end user by a multihop infrastructureless architecture through the sink (A $\rightarrow$ B $\rightarrow$ C $\rightarrow$ D $\rightarrow$ E) as shown in Figure 1. The sink may communicate with the *task manager node* via Internet or Satellite.

The protocol stack used by the sink and all sensor nodes is given in Figure 2. This protocol stack combines power and routing awareness, integrates data with networking protocols, communicates power efficiently through the wireless medium, and promotes cooperative efforts of sensor nodes. The protocol stack consists of the *application layer*, *transport layer*, *network layer*, *data link layer*, *physical layer*, *power management plane*, *mobility management plane*, and *task management plane*. Depending on the sensing tasks, different types of application software can be built and used on the application layer. The transport layer helps to maintain the flow of data if the sensor networks application requires it. The network layer takes care of routing the data supplied by the transport layer. Since the environment is noisy and sensor

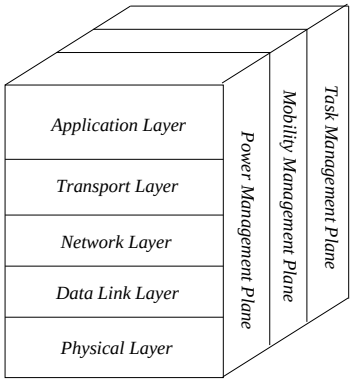

Figure 2: The sensor networks protocol stack.

nodes can be mobile, the MAC protocol must be power aware and able to minimize collision with neighbors' broadcast. The physical layer addresses the needs of a simple but robust modulation, transmission and receiving techniques. In addition, the power, mobility, and task management planes monitor the power, movement, and task distribution among the sensor nodes. These planes help the sensor nodes coordinate the sensing task and lower the overall power consumption.

1.2 Research Objectives and Solutions

The objective of this thesis is to provide three important protocols that enable QoS in sensor network applications. These protocols are in the areas of routing, time synchronization, and localization. The reasons for choosing these three areas are (1) many applications need to know the location and time of events and (2) data must be ensured during delivery, e.g., shortest time. The QoS routing (QSR) protocol resides in the network layer while the time-diffusion synchronization protocol (TDP) and distributed perceptive localization framework (PLF) are part of the application layer as shown in Figure 3. Overviews of the QSR, TDP, and PLF are described in Sections 1.2.1, 1.2.2, and 1.2.3.

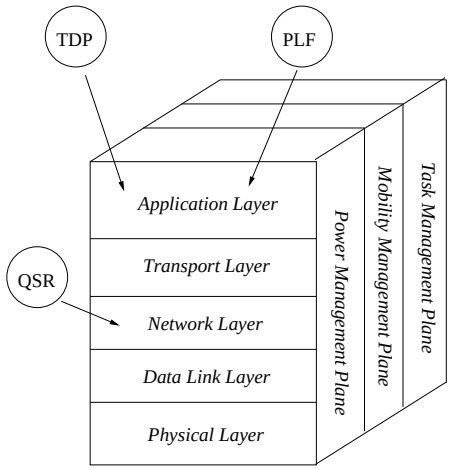

Figure 3: Schemes residing in the protocol stack.

1.2.1 Quality-of-Service Routing (QSR) Protocol

As the number of communication components that can be integrated into a single chip increases, the possibility of high volume but low cost sensor nodes is realizable in the near future. These sensor nodes may be used to collect and route information back to the sink. As a result, a bi-directional routing protocol is needed to interactively control the actuators and sensors residing on the sensor nodes. In addition, QoS such as shortest time and robustness of the route may provide applications with more data gathering and sensor node controlling capabilities. These capabilities enable the sensor networks to collect data as well as influence the environment in a distributed fashion. Hence, the QSR protocol is proposed to address these challenges. It sets up a connection based on the task assigned by the sink. The connections may be fused or aggregated to provide more efficient data gathering and sensor node controlling. In addition, the QSR protocol may be tailored for different types of traffic.

1.2.2 Time-Diffusion Synchronization Protocol (TDP)

In the near future, small intelligent devices will be deployed in homes, plantations, oceans, rivers, streets, and highways to monitor the environment. These devices require time synchronization, so voice and video data from different sensor nodes can be

fused and displayed in a meaningful way at the sink. Instead of time synchronization between just the sender and receiver during an application like in the Internet, the sensor nodes in the sensor field have to maintain a similar time within a certain tolerance throughout the lifetime of the network. The TDP is designed as a network-wide time synchronization protocol. It allows the sensor network to reach an equilibrium time and maintains a small time deviation tolerance from the equilibrium time. In addition, it is analytically shown that the TDP achieves a tolerance precision in the order of microseconds. Also, simulations are performed to validate the effectiveness of TDP in synchronizing the time throughout the network and balancing the energy consumed by the sensor nodes.

1.2.3 Distributed Perceptive Localization Framework (PLF)

The application of many sensor nodes to track the location of an event or target is appealing and has caught the attention of many researchers. The vision is to use many low-end sensor nodes to provide location information that a traditional single sensor may not be able to provide. Since the sensor nodes may be much cheaper than a traditional single sensor, the computational power, sampling rate, and circuit precision are lower causing non-Gaussian noise in range and angle estimations. In addition, sensor nodes may be deployed in areas where beacons or GPS are not applicable and the networks may be frequently jammed by environmental or manually induced noise.

The goal of PLF is to enable a node to detect and track the location of the neighboring node–allowing distributed localization. The PLF consists of a collaborative estimation technique and a particle filter. Its purpose is to address the noise and non-GPS issues in localization that are applicable to some sensor network applications. Furthermore, theoretical lower error bounds of the collaborative estimation techniques are provided, and simulations are conducted to show the validity of the

collaborative estimation techniques and position tracking via a particle filter.

1.3 Thesis Outline

The objective of this thesis is to provide some of the basic building blocks to realize the previously mentioned applications in Section 1.1.1. These basic blocks are QSR protocol, TDP, and PLF. The QSR protocol is presented in Chapter 2, and the TDP is described in Chapter 3. Furthermore, the PLF is discussed in Chapter 4. Lastly, the research contributions of the QSR protocol, TDP, and PLF are summarized, and the future research directions for sensor networks are explored in Chapter 5.

Chapter 2

QoS Routing Protocol for Sensor Networks with Heterogeneous Traffic

The QSR protocol takes into account the QoS required by the application. In addition, it allows fusion to take place along the route. It is one of the building blocks that enables QoS applications since obtaining data in an efficient way is critical. The motivation and related work of the QSR protocol is described in Section 2.1. Afterwards, the QSR protocol is presented in Section 2.2. Lastly, the performance evaluation of the QSR protocol is described in Section 2.3.

2.1 Motivation and Related Work

As the number of communication components that can be integrated into a single chip increases, the possibility of high volume but low cost sensor nodes is realizable in the near future. Each sensor node can be designed to perform a single or multiple sensing operations. For example, sensors may be used to detect temperature, seismic activity, object movement, and environmental pollution.

A family of adaptive protocols called *Sensor Protocols for Information via Negotiation* (SPIN) [32] is designed to address the deficiencies of *classic flooding* by *negotiation* and $resource-adaptation$. SPIN is a data dissemination protocol rather than a routing protocol. Also, a *directed* $diffusion$ data dissemination paradigm is proposed in [37]. The sink sends out *interest*, which is a task description, to all sensors. The task descriptors are named by assigning attribute-value pairs that describe the task. If the sources do have data for that *interest*, the data is routed along the

reverse path of interest propagation. The interest and data propagation and aggregation are determined locally. The sink has to refresh and reinforce the interest when it starts to receive data from the sources. However, this approach does not address the QoS needed by the connection between the source and sink, such as delivering data in the shortest time, power awareness of the selected route, prioritizing among different connections, or the ability to change interest for the selected sources without rebroadcasting a new interest to search for the sources again. The use of power-aware metrics in making routing decision to prolong an ad-hoc networks' life-time and its time to node failure is addressed by [64].

Such metrics are useful for sensor networks, and their concepts are applied to our new routing protocol *QoS Routing* (QSR). The QSR protocol is aimed to provide unicast as well as multicast routes to the sensor nodes in the sensor field. These routes are based on the QoS of the instruction given by the sink to address the challenges of sensor network, i.e., *low power consumption, high scalability, and high node failure.* For instance, an airport application requires delivery of *critical information* as well as support for *heterogeneous information* transfer. These requirements are addressed by the QSR protocol with the route selection capabilities that we describe later. For example, a video traffic stream may require a *time critical and loss sensitive* route as compared to a voice traffic stream that requires a *time critical but not loss sensitive* route. In addition, the airport application requires precise time and location capabilities that are described in Chapters 3 and 4. Since the routes created by the QSR protocol are bi-directional, the QSR protocol treats routes from and to sensor nodes in a similar fashion. Its contributions are as follows:

- *Periodic update of the routes is not needed in order to conserve energy.*
- *It is able to cope with topology changes due to node failures and environmental interferences.*

- *A routing table is not needed at each sensor node. As a result, memory usage is minimized at each node.*
- *It can easily incorporate new sensor nodes into the route selection process.*
- *Sources determine the routes based on QoS requirements.*
- *It allows one-to-one, many-to-one, one-to-many, and many-to-many communications.*
- *It easily incorporates lower layer topology maintenance protocols, e.g., SPAN [13], GAF [76], and LEACH [33] and higher layer protocols, e.g., data fusion and aggregation.*

The QSR protocol is described in Section 2.2. The functionalities and design concepts are explained. In addition, the performance evaluation of the protocol is discussed in Section 2.3.

2.2 QoS Routing (QSR) Protocol

The objective of the QSR protocol is to provide bi-directional routes based on the QoS required by the heterogeneous traffic: *shortest time*, *highest average energy* and *robust* routes for *(i) time critical but not loss sensitive*, *(ii) loss sensitive but not time critical*, *(iii) neither time critical nor loss sensitive*, and *(iv) time critical and loss sensitive* traffic. To achieve this objective, the design philosophy of the QSR protocol is based on ants searching for food. The ants are scattered in all directions in search of food leaving a chemical trace A along their way. The chemical trace A is for the ants to backtrack the path to the ant hive. Once the food is found, the ants backtrack to the hive leaving another chemical trace B. The intensity of the chemical trace B indicates the size of the food. If the intensity is great, more ants from the hive would help to fetch the food by following the chemical trace B to find the food.

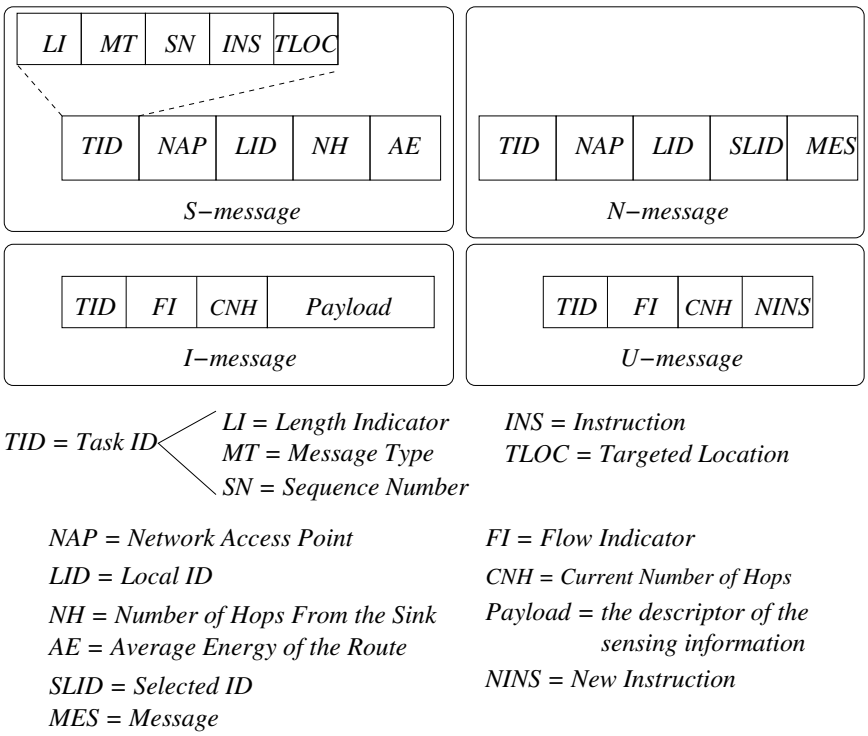

Figure 4: Messages used by the QSR protocol.

The design philosophy is mapped into the QSR protocol as follows:

- Hives ⇒ sinks
- Ants searching for food ⇒ *scout messages* (*S-messages*)
- Chemical trace A ⇒ *connection cache* (*C-cache*)
- Food ⇒ sources containing a *task cache* (*T-cache*)
- Chemical trace B ⇒ *neighbor-neighbor messages* (*N-messages*) modifying the *C-caches*
- Ants fetching the food ⇒ *information messages* (*I-messages*)
- Change condition of food fetching process ⇒ *update messages* (*U-messages*)

The sinks send *S-messages* to search for the sources. The nodes that receive the *S-messages* create an entry of the *C-cache*. The *C-cache* stores information about

the neighbor nodes for connection establishment. Once the sources are found, each source creates an entry of the *T-cache* indicating the task that the source is assigned. It determines the QoS required for the instruction and issues an *N-message*. The *N-message* backtracks to the sinks causing appropriate nodes to modify the *C-cache* to satisfy the QoS. Afterward, the *I-messages* and *U-messages* may be delivered in both directions, i.e., source-to-sink and sink-to-source. The delivery of *I-messages* and *U-messages* may be viewed as a river stream flow, because these messages are control flooded between the source and sink. Natural river stream flows, e.g., Mississippi rivers, may be merged together if they meet. Correspondingly, multiple river stream flows formed by the same or different tasks may be merged allowing data aggregation and fusion by the upper layer protocols. In addition, the width of the stream controls the redundancy and robustness of the route allowing lower layer topology maintenance protocols, e.g., SPAN and GAF, to prolong the lifetime of the sensor network. This is possible because some of the nodes may be turned OFF to conserve energy, and the stream may not be affected since there are other nodes in the stream providing the route.

The previously described design philosophy of the QSR protocol can be separated into 5 core communication and 3 maintenance procedures as described in Sections 2.2.1 and 2.2.2. The core communication procedures consist of *source discovery*, *stream selection*, *stream establishment*, *I-message transmission*, and *task termination* while the maintenance procedures include *stream reconnection*, *stream experienced sudden death*, and *instruction update*. The QSR protocol uses four types of messages, and their purposes are listed in Table 1. In addition, each message format is illustrated in Figure 4 and serves as a reference throughout the paper.

Table 1: Message types and their purposes.

Message	Purpose
S-message	Used to find the sources in the sensor field.
N-message	Used to communicate locally with neighbors for local operations.
I-message	Used to carry data information.
U-message	Used to update instruction at the sources as well as terminate the streams.

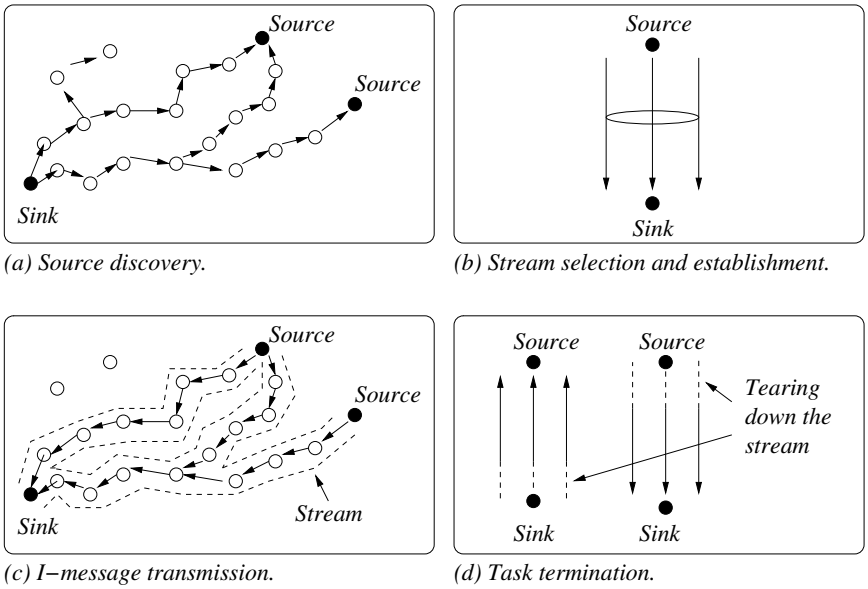

Figure 5: Overview of the QSR protocol.

2.2.1 Core Communication Procedures

A simple snapshot of the core communication procedures is illustrated in Figure 5. As shown in Figure 5.(a), the sink sends an *S-message* to find the sources. When the sources receive the *S-message*, they set up streams depending on the QoS requirement of the instruction as shown in Figure 5.(b). As illustrated in Figure 5.(c), multiple streams may be created and merged for bi-directional communications using I-message. The task termination consists of tearing down the streams as shown in Figure 5.(d). For each of the core communication procedures, a brief overview is given before a detail description is described in the following sections.

2.2.1.1 Source Discovery

Overview: The purpose of source discovery is to find the sources to execute a task. To do this, the sink sends out an *S-message*. All nodes receiving the *S-message* except the targeted nodes create an entry in the *C-cache*. On the other hand, the targeted nodes create an entry in the *T-cache.* At the end of this procedure, all nodes that have received the *S-message* has an entry either in their *C-cache* or *T-cache* regarding the task broadcast by the sink.

Procedure description: The fields of the *S-message* are illustrated in Figure 4. The TID field is the *task ID* field, which consists of four subfields, i.e., LI, MT, SN, INS, and $TLOC$. The *length indicator* (LI) indicates the length of the message. The *message type* (MT) field indicates the type of message that this packet is carrying, i.e., MT=0 stands for *S-message*; MT=1 indicates an *I-message*; MT=2 represents an *U-message*; and MT=3 corresponds to a *N-message*. The *sequence number* (SN) field allows sensor nodes to check if they have received the same message.

The *S-message* carries the task issued by the sink. The task is represented by the numeric values of the *instruction* (INS) and *targeted location* ($TLOC$) fields. For example, the sink gives the instruction "Sensor nodes detect temperature at every 10 minutes in 10 meters radius", and this instruction may be mapped to an INS value of 0. The instruction tells the sensor nodes that are within the radius of 10 meters from the location specified by the $TLOC$ field to detect the temperature at every 10 minutes.

Since each node is designed to perform specific instructions, e.g., detecting temperature, the number of instructions may be very small, and the INS values representing the instructions may be predefined and loaded into the nodes initially. The number of instructions depends on the sensor network application; similarly, the number of bits assigned to the INS field also depends if the sensor network application expects to

assign new instructions to the nodes after the nodes are deployed. For example, the sensor network application may only need 16 instructions for data collection before deployment but may want to leave some room for further instruction uploads; hence, the length of the INS field may be 5 bits long instead of 4.

To indicate where the task is originated, the *network access point* (NAP) field as shown in Figure 4 contains a value, which represents a unique sink. The number of sinks deployed is very small when compared to the number of nodes in the network. For example, there may be only 3 or 4 sinks when 4 to 5 thousands sensor nodes are in the sensor field. Since the *S-message* is routed to all sensor nodes in the sensor field, a node must be able to determine the neighbor that has sent the message. Each node in the sensor field has a *local ID* (LID) that is selected randomly from a set, which has values ranging from 1 to κ, where κ is the maximum value of the set. This way the nodes do not require a unique ID, and new nodes can be deposed off easily and deployed without the ID restriction.

The broadcast range of a node is around 1 meter to 10 meters. If the value κ is large but still small as compared to the number of nodes deployed, the probability of having the same ID within the broadcast range is rather small. In cases where the nodes have the same IDs, the streams created are not affected. For instance, this may be viewed as a river stream with a small amount of water branching off at one point and joining back the main river stream at another point. Corresponding, if two neighbor nodes have the same IDs, they may belong to the same stream. Eventually, they will merge together. It just creates redundancy for data transmission.

The *number of hops* (NH) and *average energy* (AE) fields store the total number of nodes and average energy of the route that the *S-message* has traversed prior to the current node, respectively. The following are the steps carried out by a node receiving the *S-message*.

- **Step 1:** The node checks the INS and $TLOC$ values of the *S-message* and

determines if the task is intended for it or not.

- **Step 2:** If the *S-message* is not intended for the node and the node is not receiving the same *S-message*, the following steps (2.i, 2.ii, and 2.iii) are carried out.

  - **2.i:** The node stores the SN, INS, $TLOC$, NAP, LID, NH, and AE values in an entry of the *C-cache* that represents a possible connection through the node. As a result, the entry keeps track of the downlink neighbor node that is capable of routing information back to the sink. The entry of the *C-cache* also keeps track of the uplink neighbor node if the node is selected to be part of a stream as described in Section 2.2.1.3.

  - **2.ii:** The node calculates an new average energy AE of the route and increases the number of hops NH by 1. It then inserts the new AE, new NH, and LID of the node into the AE, NH, and LID fields of the *S-message*, respectively.

  - **2.iii:** The node broadcasts the updated *S-message* to its neighbors.

- **Step 3:** If the *S-message* is intended for the node, the steps 3.i and 3.ii are carried out.

  - **3.i:** The source node keeps on listening for *S-messages* for σ seconds. Within σ seconds, there may be upto η number of *S-messages*. The *arrival time* of the j^{th} received *S-message* is represented by τ_j. The route associated with the first received *S-message* is considered the shortest delay stream while the route associated with the last received *S-message* is the longest delay stream. The received *S-messages* have the same INS, $TLOC$, and NAP fields but a different LID value. For each received *S-message*, the node stores the SN, INS, $TLOC$, NAP, LID, NH, and AE values in an entry

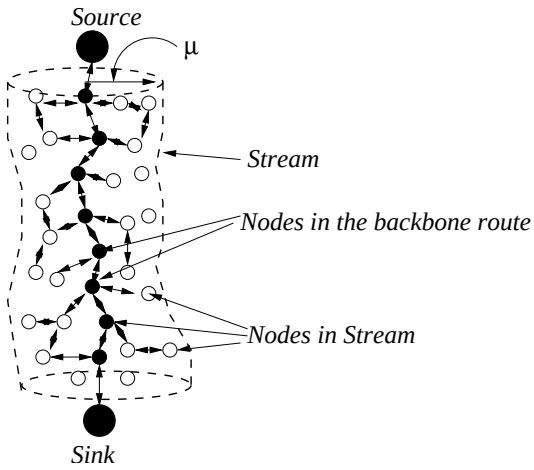

Figure 6: Stream.

of the *task cache* (*T-cache*). As a result, the entry holds information related to the task being assigned to it. In addition, it contains a *downlink sensor node problem* (*DSP*) indicator. The *DSP* indicator is used to identify if the downlink neighbor node is good to route messages or not.

– **3.ii:** After σ seconds, the node selects the type of streams to transmit the *I-message* back to the sink according to the QoS requirements of INS as described in Section 2.2.1.2.

2.2.1.2 Stream Selection

Overview: The purpose of route selection is to select the type of streams for the requested traffic. Before describing the selection process, the structure of a stream is described. At the end of the selection process, neighbor nodes are selected to form a stream or streams.

Procedure description: As illustrated in Figure 6, the structure of a stream between a source and sink is given. Each stream has a backbone route that is used to relay messages hop-by-hop using only one node at a time. The width of the stream is specified by the width μ, where μ is the number of hops away from the nodes in the backbone route. It controls the robustness of the stream by creating redundancy

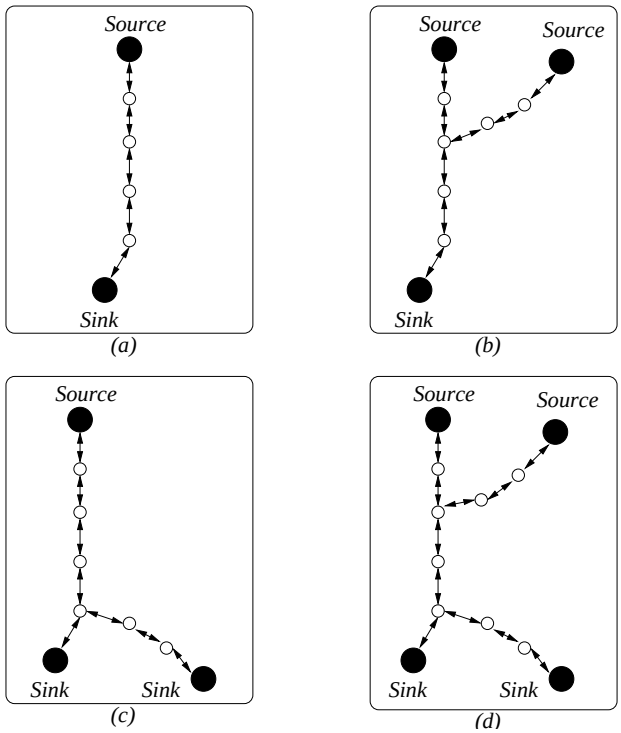

Figure 7: Different scenarios of streams: (a) single source and sink, (b) multiple sources and single sink, (c) multiple sink and single source, and (d) multiple sources and multiple sinks.

but pays the price of higher energy consumption. The nodes in the backbone route always relay incoming messages while the other nodes in the stream assist, just in case messages are lost.

A stream may merge with other streams and diverge to multiple sinks as shown in Figure 7. There are four scenarios that streams may interact. The first scenario illustrated by Figure 7.(a) shows a stream between a pair of source and sink nodes. If there are more than one source node, the streams may be merged together if they meet somewhere between the sources and the sink as shown in Figure 7.(b). The portion where the streams are merged relies on higher layer protocols to aggregate and fuse the messages since the data rate increases when streams are merged. The streams may also diverge to multiple sinks if the messages are intended for multiple sinks as shown in Figures 7.(c) and 7.(d).

After σ seconds since the first reception of the *S-message* at the sources, the

sources determine the QoS required for the instruction being assigned. *There are 4 types of streams with each corresponding to a type of traffic that the sources may establish.* Below is a list of the types and its associated actions carried out by the sources. The rationale in selecting the type of streams is based on four parameters: the arrival time of the j^{th} received *S-message*, τ_j; the average energy of the route, AE; the number of possible streams, η; and the width of each stream, μ. By adjusting these four parameters, the source may choose a different type of streams for different scenarios. For example, a video feed may be delivered by a type 4 stream while a voice signal may be delivered by a type 1 stream. In summary, each type of stream is a tradeoff among delay, average energy, and robustness.

1. **Type 1: Carry time critical but not loss sensitive data.**
   Actions: (1) Choose LID with the lowest arrival time of the *S-message*, i.e., τ_1. (2) Choose the stream width μ.

2. **Type 2: Carry loss sensitive data that is not time critical.**
   Actions: (1) Choose η LIDs with the highest AE. (2) Choose the stream width μ.

3. **Type 3: Carry data that is neither time critical nor loss sensitive.**
   Actions: (1) Choose the LID with the highest AE. (2) Choose the stream width μ.

4. **Type 4: Carry data that is time critical and loss sensitive.**
   Actions: (1) Choose η LIDs with the lowest arrival time of the *S-message*, i.e., τ_1 ... τ_η. (2) Choose the stream width μ.

The above actions specify how to select the neighbors to form a stream or streams. The types 1 and 3 each has one stream while the types 2 and 4 each may have upto η streams. In certain situation, a source may be instructed to perform the same instruction by multiple sinks. For such scenario, the source uses the same neighbor node to route the data back to the sinks. This way only one copy of the data is sent.

In summary, the different types of streams are presented in Table 2. For example, $S_2(\eta, \mu)$ represents η number of type 2 streams; each has width μ.

Table 2: Different types of stream.

Type 1	**Type 2**	**Type 3**	**Type 4**
$S_1(1, \mu)$	$S_2(\eta, \mu)$	$S_3(1, \mu)$	$S_4(\eta, \mu)$

2.2.1.3 Stream Establishment

Overview: The purpose of stream establishment is to create the stream that is chosen. The source sends an *N-message* to the downlink node, and nodes that are within μ hops away from the downlink node are part of the stream. This process repeats hop-by-hop until the *N-message* reaches the sink. At the end, the *C-cache* entries of all the nodes that are part of the stream are modified while the entries of other nodes are deleted.

Procedure description: After the type of streams is selected as described in Section 2.2.1.2, the source sends an *N-message*. The MES of the *N-message* is set equal to a `new connection message with value` μ indicator. The INS, $TLOC$, and NAP values of the *N-message* are the same as the *S-message* that the source is responding to. The LID field of the *N-message* is set equal to the LID value of the source, and the $SLID$ value is set equal to the downlink LID value that is stored in the entry of the *T-cache*. If the source chooses η number of streams, then η *N-messages* are broadcast by the source; each message is targeted at different downlink LID.

After the broadcast, the neighbor nodes receive and check if the INS, $TLOC$, and NAP values match the ones stored in an entry of the *C-cache*. If a match is found, the nodes extract and compare the $SLID$ value in the *N-message* with their LID value. There are three scenarios that can happen:

- **Scenario 1:** If the $SLID$ value matches the LID value of the node, the node sets

the uplink LID value in the corresponding entry of the *C-cache* equal to the value stored in the LID field of the *N-message*. In addition, the *Node Selected* indicator is set to TRUE and the value μ from the *N-message* is stored in the entry. Afterwards, the node broadcasts a new *N-message* with LID and $SLID$ values set equal to the LID of the node and downlink LID from the corresponding entry of the *C-cache*, respectively. The TID, NAP, and MES values in the *N-message* stays the same as the ones that are received. This procedure creates the backbone route as illustrated in Figure 6. If during this procedure that the MAC layer indicates the *N-message* is lost due to possible node failure or congestion, the node in the backbone route may retransmit the *N-message* upto ϵ times, i.e., the retransmission limit.

- **Scenario 2:** If the $SLID$ value does not match the LID value of the sensor nodes and the value μ specified by the MES field of the *N-message* is greater than 0, the nodes then set the *Node Selected* indicator to TRUE in the corresponding entry of the *C-cache*. These sensor nodes rebroadcast the *N-message* with $SLID$ set equal to 0 and μ value decreased by 1. Sensor nodes receiving the same *N-message* but with different μ value do not rebroadcast. They only rebroadcast when they first receive the *N-message*. When the μ value is decreased to 0, the sensor nodes stop the rebroadcast of the *N-message*. As a result, only sensor nodes that are μ hops away from the nodes of the backbone route form a stream. A node in the backbone route may rebroadcast the *N-message* twice; once when it receives the *N-message* from another node in the backbone route and twice from another node in the stream that is not part of the backbone route.

- **Scenario 3:** If the $SLID$ value does not match the LID value of the sensor nodes and the μ value in the *N-message* is equal to 0, the nodes delete the entry from the *C-cache* that has the same INS, $TLOC$, and NAP values as in the

N-message. As a result, sensor nodes that are not part of a stream remove the entry associated with the *N-message* from the *C-cache*.

Once the *N-message* has reached the sink, the chosen stream is established. At this point, the source may send *I-messages* or *U-messages* to the sink. If the nodes in the sensor field have not received an *N-message* after receiving the *S-message* in ζ seconds, the sensor nodes delete the entry from the *C-cache* that was created when the *S-message* was received. The nodes free up the memory, so they can create entries in the *C-cache* for other incoming *S-messages*. The value ζ depends on the size of the sensor field.

2.2.1.4 I-message *Transmission*

Overview: The purpose of *I-message* transmission is to transmit *I-messages* through the established stream. The *I-messages* only have to specify the task and the travelling direction. The task is represented by the INS and $TLOC$ values while the traveling direction is by the FI value. The *I-messages* may travel either toward the sink or source.

Procedure description: The fields of the *I-message* are illustrated in Figure 4. The TID field of the *I-message* contains the same INS and $TLOC$ values as the ones in the received *S-message*, so neighbor sensor nodes can determine if they are responsible to route the *I-message*. The FI field is only 1 bit long, which is used to indicate if the message is routed toward the source (FI=1) or the sink ($FI = 0$). The CNH field contains the NH value that is stored in the entry of either the *T-cache* or *C-cache*. The source obtains the NH value from the entry of the *T-cache* while the other nodes in the stream obtain it from the *C-cache*. The cache entry has the same INS and $TLOC$ values as in the *I-message*. Lastly, the *Payload* field of the *I-message* contains a descriptor, e.g., a temperature value, voice clip, or video frame, of the sensing information.

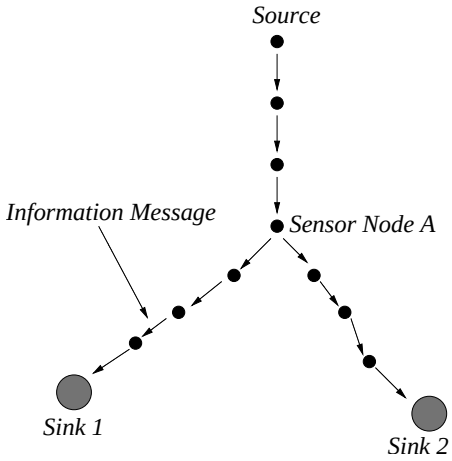

Figure 8: First part of the streams shared.

When a source or sink broadcasts an *I-message*, a node receiving the message rebroadcasts only once to avoid sending the same message. Afterwards, it turns OFF the receiver for an amount of time if the sleep mode operation is ON; otherwise, the receiver stays ON. The reason for turning OFF the receiver is to avoid listening to the neighbors broadcasting the same *I-message* that the node is not interested. The duration of the time is a design parameter that trades off sensitivity for energy consumption.

There are two traveling directions of the *I-message*: (i) toward the sink and (ii) toward the source. When the *I-message* is going toward the sink, the node in the stream checks to see if the NH value stored in the entry of the *C-cache* is smaller than the CNH value of the message. If the value is smaller, then the message is forwarded after replacing the CNH value of the message with the NH value obtained from the entry of the *C-cache*. When the *I-message* is going toward the source, it is forwarded when the NH value stored in the entry of the *C-cache* is greater than the CNH value of the received *I-message*. Before forwarding the *I-message*, the CNH value of the message is replaced with the NH value stored in the entry of the *C-cache*.

Two scenarios of how shared streams relay messages are illustrated in Figures 8 and 9. As illustrated in Figure 8, the neighbor nodes at the downlink of sensor node

A route the *I-message* along their own path once sensor node A broadcasts it. On the other hand, the *I-messages* from both sources as shown in Figure 9 may be fused or aggregated at sensor node B.

2.2.1.5 Task Termination

Overview: The purpose of task termination is to tear down the stream by using an *U-message*. The *U-message* may be issued either by the sink or source. At the end, the entries of the *C-cache* are removed. In addition, entries of the *C-cache* are refreshed every θ seconds. Unused entries are removed.

Procedure description: The fields of the *U-message* are shown in Figure 4. It contains the TID, FI, CNH, and $NINS$ fields. The INS and $TLOC$ subfields of the TID field are the same as the ones used by the *S-message* to establish the stream at the beginning. A node receiving an *U-message* uses the INS and $TLOC$ values to determine if it should route the message. The FI field is used to indicate if the message is going toward the source or sink similar to the *I-message* as described in Section 2.2.1.4. The CNH field contains the NH value that is stored in the entry of the *C-cache* of the broadcasting node; the entry is associated with the task specified in the TID field. The $NINS$ contains the new instruction for the sources.

There are two situations when a task at the sources is terminated. The first situation is when the sources have finished the task given by the sink. The sources broadcast an *U-message* with $NINS$ field set and mapped to a `task completed` indicator. As this *U-message* is routed to the sink, the streams are teared down by removing the entries in the *C-cache* and *T-cache* that are associated with the task. There may be multiple entries for the same task, each originated from a different sink. All these entries will be removed when a node receives the task completion *U-message*. The *U-message* is routed using the task and traveling direction indicator. It does not specify the initiator of the task, so multiple entries in the *C-cache* may

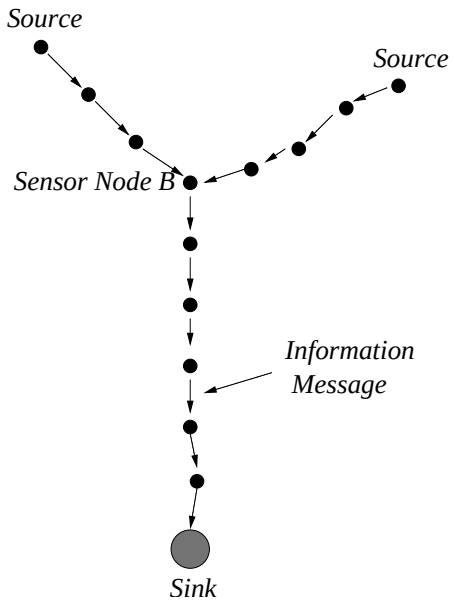

Figure 9: Bottom part of the streams shared.

be removed at once.

The second situation is when the sink decides to terminate the task. The sink sends an *U-message* with $NINS$ value set and mapped to a `task terminated` indicator. The streams are teared down as the *U-message* is routed; if a node has multiple entries containing the same task, the node does not delete any of the entries, and it also stops forwarding the *U-message*. This is to ensure that the tear down process does not break any stream that other sinks have initiated.

The entries of the *C-cache* are refreshed every θ seconds, and they are removed if they are not used within this time-frame to conserve memory usage. The value θ is a design parameter; it depends on the application of the sensor network. The application may trade off memory usage for sensitivity.

2.2.2 Maintenance Procedures

There are three maintenance procedures: *stream reconnection*, *stream experienced sudden death*, and *instruction update*. The stream reconnection procedure discusses the steps required for reconnecting a stream. If a stream is suddenly terminated,

the application may request another stream in the stream experienced sudden death procedure. The last procedure allows the application to update the instruction at the sources. These procedures are discussed in the following sections.

2.2.2.1 Stream Reconnection

Overview: The stream reconnection procedure has 3 phases: *failure detection*, *reconnection*, and *parameter update*. When a sensor node detects that it is failing, it initiates the reconnection phase to reconnect the stream by sending out an *N-message*. Afterwards, the NH parameters of the *C-caches* are updated, so messages can be routed through the stream.

Procedure description:

Failure detection: During transmission of information, a sensor node in the backbone route may determine that it is low in energy for routing or there are high environmental noises around the node as shown in Figure 10. After such decision, the node broadcasts an *N-message* with the MES field set equal to the `reconnect message` indicator. The LID and $SLID$ fields of the *N-message* are set equal to the uplink and downlink LID values, respectively, that are stored in the entry of the *C-cache*. If there are multiple entries in the *C-cache* of the affected node, a new *N-message* is broadcast for each of the entries.

Reconnection: Once a neighbor node receives the *N-message*, it checks the entries of the *C-cache* and determines if one of the entries has the same INS, $TLOC$, and NAP values as in the *N-message*. If there is a match, the neighbor node then checks to see if the LID value of the node matches either the LID or $SLID$ value of the *N-message*. If the LID value of the node matches the LID value of the *N-message*, the node is the uplink node of the node C that has problem in routing information as illustrated in Figure 10. If the LID value of the node matches the $SLID$ value of the *N-message*, the node is the downlink node. The uplink and downlink nodes are

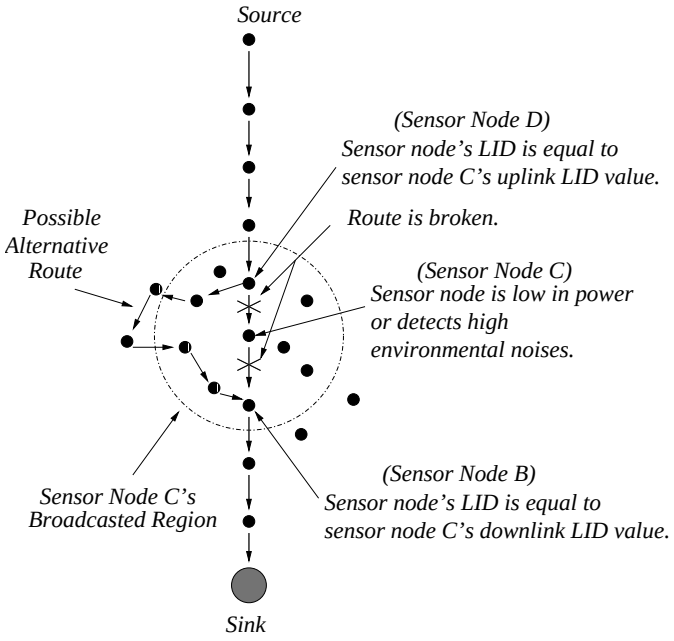

Figure 10: Reconnecting a stream.

the sensor nodes D and B, respectively, as shown in Figure 10. The neighbor nodes do not rebroadcast the *N-message*, because the message is intended for the sensor nodes D and B. The sensor node D sets the DSP indicator to TRUE in the entry of the *C-cache* and waits for an *N-message* from the sensor node B. On the other hand, the sensor node B sets the *uplink sensor node problem* (USP) indicator to TRUE and broadcasts a new *N-message* with the LID and $SLID$ fields set equal to the LID values of the sensor nodes B and D, respectively. The MES field is set equal to the `reconnect the route with value` ψ indicator, where ψ is the number of hops allowed for this message. If the sensor node D is not found within ψ hops, then the stream may be considered as having a sudden death and dealt with accordingly as explained in Section 2.2.2.2. Alternatively, the sensor node B may also increment the value ψ until the sensor node D is found, i.e., just like the ring search in ad hoc network.

After the node B has broadcast the *N-message*, any neighboring node that receives the message checks to see if the message is targeted for it. The message is targeted

for it when the LID value of the node matches the $SLID$ value of the *N-message* and the DSP indicator in the entry of the *C-cache* that has the same INS, $TLOC$, and NAP values as in the *N-message* is TRUE. If the message is not intended for the node, it rebroadcasts the *N-message* after replacing the LID value of the message with the LID value of the broadcasting node and decreasing the value ψ specified in the MES field by 1. In addition, the node creates a new entry in the *C-cache* with the values in the SN, INS, $TLOC$, NAP, and LID fields of the *N-message*. The entry is created in the same way as if the node has received an *S-message* as described in Section 2.2.1.1 except that the average energy and the number of hops from the sink are not calculated and used.

When the uplink sensor node D receives the *N-message* as shown in Figure 10, it updates the downlink LID value with the LID value of the received *N-message*. The updated value is stored in the entry of the *C-cache* that has the same INS, $TLOC$, and NAP values as in the *N-message*. The sensor node D also sets the DSP indicator of the corresponding cache entry to FALSE. If it receives more than one copy of the same *N-message*, it uses the first received *N-message*; as a result, a minimum delay stream is selected.

Afterward, the sensor node D broadcasts a new *N-message* with LID and $SLID$ fields set equal to the LID value of the node and the updated downlink LID value, respectively. In addition, it sets and maps the MES value to a `new connection message with value` μ indicator, where μ is obtained from the entry of the *C-cache* that has the same INS, $TLOC$, and NAP values as in the *N-message*. When a neighbor node receives the *N-message*, it checks to see if it is selected to be part of the stream and rebroadcasts the *N-message* as described in Section 2.2.1.3. Once the *N-message* has reached the sensor node B, the stream is reconnected between the sensor nodes D and B as shown in Figure 10.

Parameter update: The sensor node B sets the USP indicator to FALSE and broadcasts an *N-message* targeted for the uplink node with the MES value set equal to the `update NH with value` γ indicator. The value γ is equalled to the NH value in the entry of the *C-cache* that has the same INS, $TLOC$, and NAP values as in the *N-message*. When the uplink node receives the *N-message*, it sets the NH value in the corresponding entry of the *C-cache* equal to the value γ. Afterwards, the node rebroadcasts the *N-message* targeted for its uplink node with the value γ incremented by 1. This process repeats until the *N-message* has reached the source.

By the time the message has reached the source, the NH values have been updated in all the nodes that are between the sensor node B and the source. The purpose of this process is to update the NH values allowing messages to be routed toward the sink or the source as discussed in Section 2.2.1.4. If the stream reconnection procedure happens often or the stream has been setup for a long time, it is better to tear down the stream as described in Section 2.2.1.5 and initiate a new stream. This way a new stream that fits the QoS of the instruction better is used. The duration before a new stream is initiated is a system parameter depending on the environmental condition as well as the sensor node deployment strategy. For example, if new nodes are incrementally deployed, the QSR protocol may easily incorporate the new nodes into the routing process and exploit the higher node density to provide better QoS.

2.2.2.2 Stream Experienced Sudden Death

Overview: During *I-message* transmissions, a stream may experienced sudden death. The ways to handle sudden death are discussed in the following, ranging from increasing QoS to initiating a new stream.

Procedure description: There is also another scenario which affects the delivery of *I-message* from the sources to the sink. Such scenario is when the stream suddenly terminates and experiences sudden death. If the sink does not get the *I-message* at

the time when it expects, the sink sends out a new *S-message* with a higher QoS requirements version of the same instruction, i.e., a higher QoS INS value. By doing this, new streams can be established to avoid trouble spots experienced by the stream which suddenly terminates. Also, if the instruction previously requires only one stream to be established, multiple streams with a higher stream width may be established, because the QoS requirements are stricter than before. If the environment is known to inflict sudden death easily, the QoS requirements of the instruction should be stricter at the beginning. As a result, multiple streams with a higher stream width may be set up between the sources and sink to enhance the robustness of the *I-message* delivery.

2.2.2.3 Instruction Update

Overview: When an application requires different types of data, the sink may request to change the instruction at the sources.

Procedure description: The *U-message* as shown in Figure 4 allows the sink to update its instruction to the sources. From the previous example as described in Section 2.2.1.1, "Sensor nodes detect temperature at every 10 minutes in 10 meters radius" may be updated to "Sensor nodes detect temperature at every 1 minutes in 10 meters radius". The new instruction is contained in the $NINS$ field of the *U-message*. The *U-message* is routed toward the source with the FI field set equal to 1. To specify which stream to use, the task, i.e., INS and $TLOC$, is set equal to the one characterized by the stream.

2.3 Performance Evaluation

Performance evaluation is done by comparative analysis and simulation in Sections 2.3.1 and 2.3.2, respectively.

2.3.1 Performance Evaluation by Comparative Analysis

The complexity and robustness of the QSR, DSR, and Flooding protocols are compared in Sections 2.3.1.1 and 2.3.1.2, respectively. Since the QSR protocol provides unicast and multicast routes, it is compared to the DSR protocol but not the directed diffusion, SPIN1, and gossiping protocols. The directed diffusion, SPIN1, and gossiping protocols strictly focus on data dissemination while the QSR protocol focuses on providing unicast and multicast routes. As a result, we believe it is better compared to DSR protocol although directed diffusion, SPIN1, and gossiping protocols are good for data dissemination. We do compare with the Flooding protocol, which serves as a basis for the comparison between the QSR and DSR protocols as described in Section 2.3.2.1.

2.3.1.1 Complexity of the Protocols

As shown in Table 3, the complexity of the protocols is given as the number of messages required to send gathered data for one session. The QSR, DSR, and Flooding protocols require $n + (\delta + 2)\ \rho k_1$, $\rho\ (2\ n + \delta k_2)$, and $n + \rho n \delta$ number of messages, respectively. The parameters n, δ, ρ, k_1, and k_2 are defined in Table 3. The Flooding protocol requires the most amount of messages, because it sends the data messages to all the nodes in the sensor field. When the number of messages required to send the gathered data is large, the DSR protocol outperforms the QSR protocol if the stream formed by the QSR protocol uses many nodes for routing. For example, if $\rho = 100$, $\delta = 1000$, $n = 1000$, $k_1 = 40$, and $k_2 = 10$, the Flooding protocol requires 100 million messages. On the other hand, the DSR and QSR protocols require 1.2 million and 4.0 million messages, respectively. If the number of nodes per stream (k_1) is decreased from 40 to 10, the QSR protocol only requires 1.0 million messages, which is less than the DSR protocol. In addition, the QSR protocol allows the I-messages to be aggregated. As a result, the number of messages required to send gathered data

can be further reduced. For instance, if the data aggregation efficiency is 70%, the number of messages is reduced to 0.7 million.

Table 3: Number of messages per data gathering session; ρ = number of pairs of sink and sources; δ = number of packets needed to send gathered data; n = number of nodes deployed; k_1 = number of nodes within one stream; and k_2 = number of nodes within one route.

Protocols	**Description**	**Max. Number Of Messages**
QSR	Source discovery: n S-messages Stream establishment: ρk_1 N-messages I-message transmission: 0 to $\delta\rho k_1$ I-messages Task termination: ρk_1 U-messages	$n + (\delta + 2)\ \rho k_1$
DSR	Route request: ρn Route request + route reply (piggybacked) : ρn Data message transmission: $\rho\delta k$	$\rho\ (2\ n + \delta k_2)$
Flooding	Discover sources : n Data message transmission: $\rho n\delta$	$n + \rho n\delta$

2.3.1.2 Robustness of the Protocols

As described in Section 2.3.1.1, the QSR protocol is efficient in gathering data. It can also be tailored to address the frequent node failure problem. A route from the source to the sink is broken when one or more sections of the route are down. For example, if the network is partitioned, the flooding protocol can not be used to deliver the messages. The number of nodes within a section for Flooding is n/h, where n is the number of nodes deployed in the sensor field, and h is the number of hops between the source and sink. All the sensor nodes are assumed evenly distributed throughout the sensor field. As for the DSR and QSR protocols, the DSR protocol has only one node while the QSR protocol has k_1/h number of nodes. The node failure probability is given by α. As a result, the probabilities of a section being down for the QSR, DSR, and Flooding protocols are as follows: $\alpha^{k_1/h}$ (QSR), α (DSR), and $\alpha^{n/h}$ (Flooding).

The route failure probabilities for the Flooding, DSR, and QSR protocols are given in Table 4. They are the sum of different section failure probabilities. The

Table 4: Route failure probability; h = number of hops between source and sink; α = node failure probability; k_1 = number of nodes within one stream; and n = number of nodes deployed.

Protocols	**Route Failure Probability**
QSR	$\sum_{i=1}^{h} \alpha^{i \cdot k_1/h}$
DSR	$\sum_{i=1}^{h} \alpha^{i}$
Flooding	$\sum_{i=1}^{h} \alpha^{i \cdot n/h}$

route failure probability for the QSR protocol is between the DSR and Flooding protocols. If the number of nodes within one stream k_1 is equal to the number of hops h, the route failure probability is equal to the failure probability of the DSR protocol. On the other hand, if the stream increases its width μ, the route failure probability approaches the failure probability of the Flooding protocol. As a result, the QSR protocol can be tailored for different types of sensor networks.

2.3.2 Performance Evaluation By Simulation

The performance of the QSR protocol is also evaluated with an event driven simulation. The performance data is collected from 100 simulation runs; each run is 1800 seconds with a message sent at every 10 seconds. One thousand non-mobile sensor nodes is deployed randomly in a 200 meters by 150 meters sensor field. Each of the sensor nodes can receive and transmit messages to its neighbors by executing the routing protocol independently, i.e., each sensor node is emulating a physical sensor node where it has its own memory and routing state.

For the performance evaluations, the sink and source nodes are located at (0,0) and (180,130) of the sensor field. The configuration of each sensor node is listed in Table 5, which has the parameters as in [32] but with energy randomly distributed from 5K to 10K Joule.

As specified in Table 5, the *processing delays* are 0.05 seconds and 0.01 seconds for saturated and not saturated sensor network, respectively. They are composed of

Table 5: Configuration of each sensor node.

Parameters	**Value**
Transmission radius	10 meters
Available energy	5K to 10K Joule
Transmission cost	600 mW
Receiving cost	200 mW
Idle cost	200 mW
Transmission rate	1 Mbps
Signal propagation speed	$3 * 10^8$ meters/second
Processing delay (saturated)	0.05 seconds
Processing delay (not saturated)	0.01 seconds
Queueing delays	none

delays incurred at the lower layers, i.e., medium access and physical layers. For example, a sensor node, which is equipped with an 802.11 MAC, may have a processing delay of around 0.01 seconds when the network is not saturated [44]; the delay value does not change for different node densities, e.g., 5, 10, and 20 nodes, and it is near constant until the network becomes saturated at around 75% of channel capacity. At saturation, the processing delay of an outgoing message, the number of retransmission, and the end-to-end delay flatten at different values [44][72][18][7] depending on the 802.11 MAC parameters, e.g., node density and congestion window size. The processing delay for a saturated sensor network is assumed to be 0.05 seconds [44].

The QSR protocol is compared to the Flooding [32] and DSR [39] protocols in Section 2.3.2.1. The Flooding protocol does not require a node to have a unique ID in order to identify the neighbors of the node, i.e., the maximum number of IDs assigned to sensor nodes is equal to the number of nodes deployed. On the other hand, the DSR protocol does require a unique ID, because it needs to know the exact neighbor that the message is intended. As for the QSR protocol, it only uses 800 IDs when deploying 1000 nodes in all the simulation runs. A more in-depth analysis of the QSR protocol is discussed in Section 2.3.2.2.

The following is a table listing the length of each message used in different protocols.

Table 6: List of messages and its length used in different protocols.

Protocols	**Message And Its Length In Bits**
QSR	*N-message*, *S-message*, and *U-message* are 80 bits; *I-message* is 4000 bits.
Flooding	The data message is 4000 bits.
DSR	The request, reply, error, and data messages are 48, 32, 64, and 4000 bits, respectively. (The message sizes start with these values and change depending on the number hops required.)

2.3.2.1 Comparison of different protocols

When comparing different protocols, the sensor network is assumed to be saturated with a processing delay of 0.05 seconds as listed in Table 5. In addition, there will be node failures and packet losses. This is to simulate heavy background traffics due to multiple sources and other data gathering operations. Also, the QSR protocol sets the retry limit ϵ to 10 when sending a *N-message* during the stream establishment procedure. This is to improve the chance of creating a stream between the source and sink.

The throughput of the Flooding, DSR, and QSR protocols as the node failure and packet loss probabilities increase is illustrated in Figures 11 and 12. It is defined as the ratio of the number of successive transmissions over the number of attempted transmissions before a connection is broken or terminated. As the node failure and packet loss probabilities increase, the throughput of the DSR protocol drops significantly and performs worse than the QSR protocol. The reason for this performance drop is because the message contains the route between the source and sink in its header. If any of the nodes in the route is down or can not receive the message, an error message has to be sent back to the source to update the route. Since the

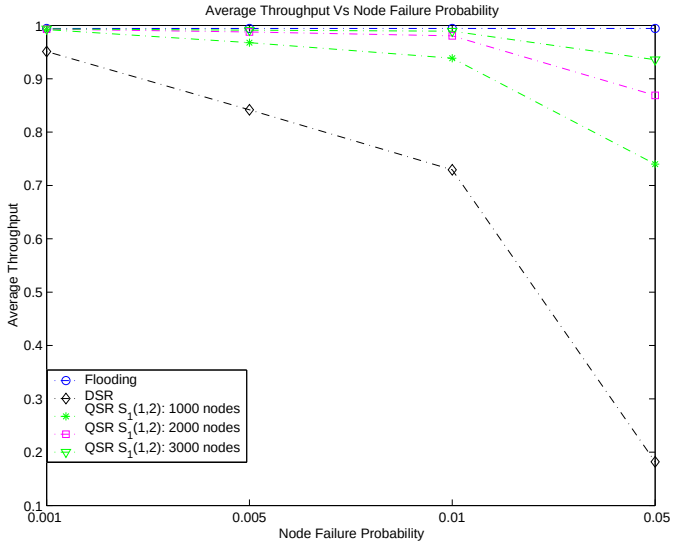

Figure 11: The throughput when node failure increases.

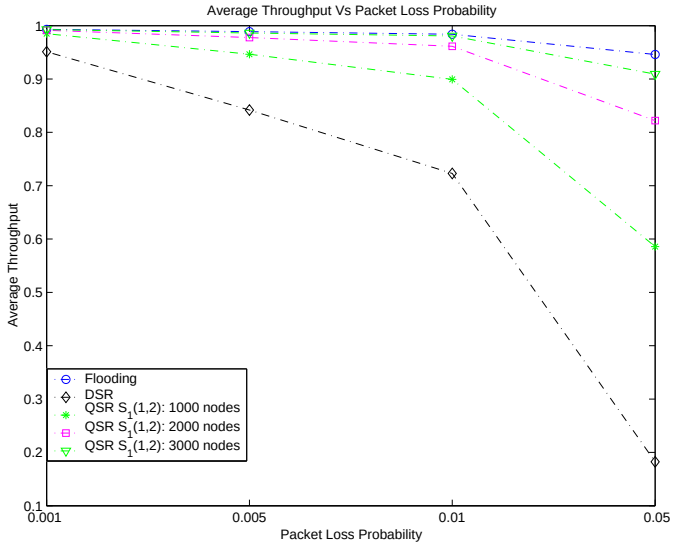

Figure 12: The throughput when packet loss increases.

number of hops is large, a broken route happens more often with the DSR protocol. This effect is captured by Figures 11 and 12. The throughput of the DSR protocol is near 0.2 when the node failure and packet loss probabilities are 0.05.

The stream $S_1(1, 2)$ used by the QSR protocol is specifically designed to increase the robustness of the route between the source and the sink since the stream has a wide width as described in Section 2.2.1.2. In addition, the stream takes advantage of the

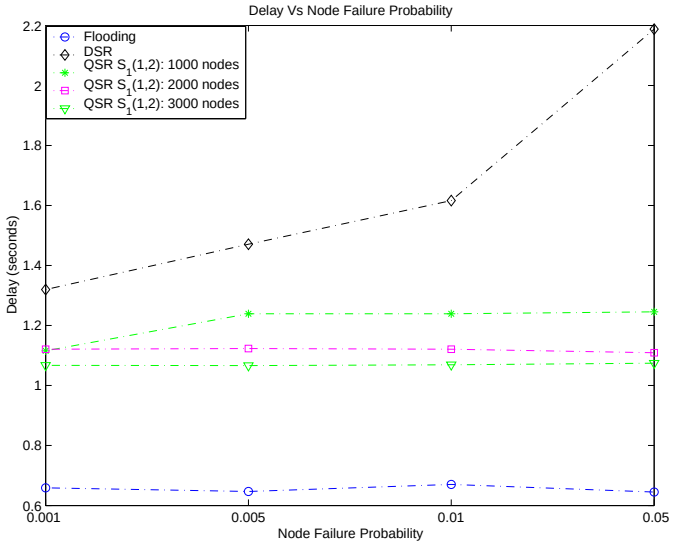

Figure 13: Delay between the source and sink as node failure increases.

increased node density. As shown in Figures 11 and 12, the throughput increases as the number of deployed nodes increases from 1000 to 3000. As expected, the Flooding protocol has a throughput value near 1, which means that almost all messages are received by the sink when they are sent by the source.

The performance of the DSR, QSR, and Flooding protocols matches our theoretical robustness analysis in Section 2.3.1.2. The route failure probability of the QSR protocol is between the DSR and Flooding protocols. As shown in Figures 11 and 12, the throughput of the QSR protocol is between the throughput of the Flooding and DSR protocols. Also note that the throughput of the QSR protocol increases as the number of nodes increases. This corresponds to the decrease of the route failure probability as k_1 increases, which is the number of nodes within the stream.

One other important characteristic of a routing protocol is the delay between the sink and source. The performance of this characteristic is illustrated in Figures 13 and 14. The DSR protocol is very sensitive to node failures and packet losses; as the node failure and packet loss probabilities increase, the average delay increases to around 2.2 seconds for both the node failure and packet loss scenarios as shown in Figures 13 and 14. On the other hand, the Flooding and QSR protocols are not as

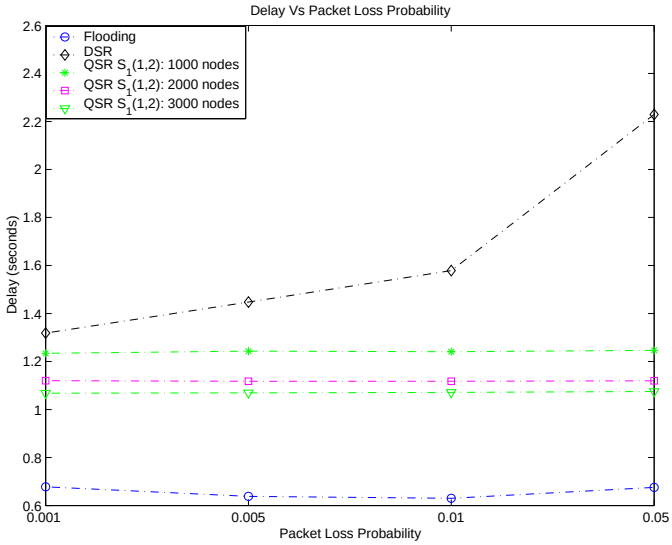

Figure 14: Delay between the source and sink as packet loss increases.

sensitive to the increase of node failures and packet losses as the DSR protocol; out of these two protocols, the QSR protocol has a larger delay, but it has the smallest jitter out of the three protocols as shown in Figures 15 and 16. The jitter is defined as the standard deviation of the delay. In addition, the jitter for the QSR protocol is not affected by the node failures and packet losses.

In summary, the QSR protocol is even effective when the node failures and packet losses are high; it is able to exploit the high density nature of the sensor networks; as the number of nodes deployed increases, the throughput of the QSR protocol increases creating more resilience to node failures and packet losses. In addition, the delay and jitter are not affected much as the node failures and packet losses increase.

2.3.2.2 In-depth performance evaluation of the QSR protocol

As stated in Section 2.2.1.2, there are four types of traffics that the QSR protocol can support. To show that the QSR protocol can support the different characteristics of the types, a lumped model of the available streams between the source and sink is illustrated in Figure 17. The characteristic of each lumped stream has an average energy AE and delay ranging from 2K Joule to 8K Joule and 1.0 second to 1.3 seconds

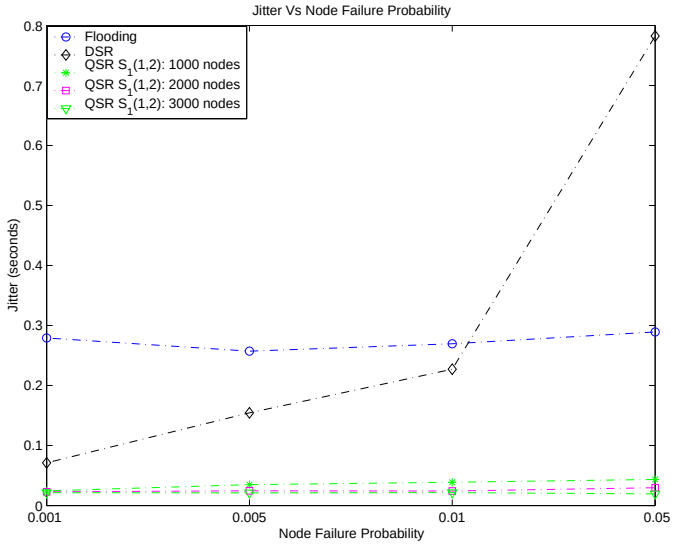

Figure 15: Jitter as node failure increases.

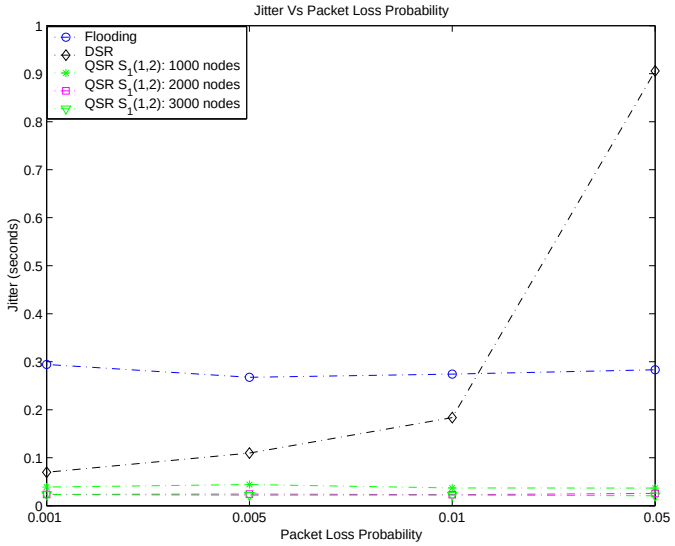

Figure 16: Jitter as packet loss increases.

(observed from Section 2.3.2.1), respectively. In addition, each stream has packet loss probability of 0.050.

The source picks one of the three streams for type 1 and 3 traffics while two of the three streams for type 2 and 4 traffics according to the rules in Section 2.2.1.2. The average delay, jitter, and packet loss probability are obtained from simulations using the lumped model in Figure 17. The average delay and jitter from the source to sink

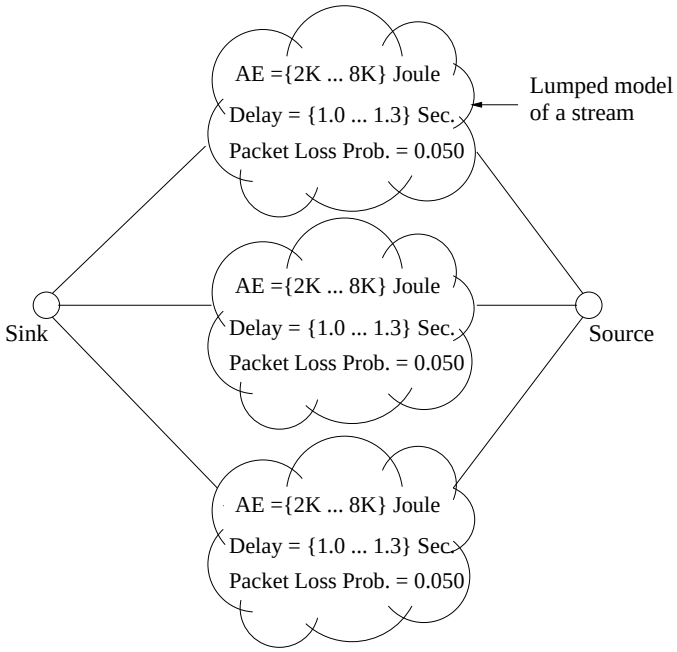

Figure 17: The lumped model of different streams.

and the average packet loss probability for each type of traffics are given in Table 7. The type 1 traffic, which is intended for time critical but not loss sensitive data, has the lowest delay (1.056 seconds) and jitter (0.075 second) while it has the highest packet loss probability of 0.050. Since type 2 traffic is intended for loss sensitive data and can tolerate a higher delay, the packet loss probability is 0.015. As for type 3 traffic, it has the highest delay, jitter, and packet loss probability since the type 3 traffic is both not time critical and loss sensitive. Lastly, the type 4 traffic is meant for both time critical and loss sensitive data. It has the lowest packet loss probability of 0.014 while the delay and jitter are second to the type 1 traffic.

Table 7: Delay and packet loss probability of different streams.

Stream Type	**Delay (Seconds)** Mean/Jitter	**Packet Loss Probability** Mean
Type 1	1.056/0.075	0.050
Type 2	1.126/0.101	0.015
Type 3	1.150/0.112	0.050
Type 4	1.074/0.079	0.014

Chapter 3

Time-Diffusion Synchronization Protocol for Sensor Networks

Another important building block besides the QSR protocol as described in Chapter 2 is time synchronization. The motivation and related work for designing a time synchronization protocol is given in Section 3.1. Afterwards, the design issues and system architecture of the time synchronization protocol are discussed in Section 3.2. The discussion is followed by a detail description of the time synchronization protocol in Section 3.3. Lastly, analytical performance evaluation and simulations are provided in Sections 3.4 and 3.5, respectively.

3.1 Motivation and Related Work

In the near future, small intelligent devices will be deployed in homes, plantations, oceans, rivers, streets, and highways to monitor the environment [2]. Events such as target tracking, speed estimating, and ocean current monitoring require the knowledge of time between sensor nodes that detect the events. In addition, sensor nodes may have to time-stamp data packets for security reasons. With time synchronization, voice and video data from different sensor nodes can be fused and displayed in a meaningful way at the sink.

Instead of time synchronization between just a sender and a receiver for a particular application as in the Internet, the sensor nodes in the sensor field must maintain an equal time within a certain tolerance throughout the lifetime of the network. Combining with the criteria that sensor nodes have to be energy efficient, low-cost, and small in a multi-hop environment, this requirement becomes a challenging problem

to solve. In addition, the sensor nodes may be left unattended for a long period of time, e.g. in deep space or on an ocean floor. Note that in short distance multi-hop broadcast, the data processing time and its variation contribute the most to time fluctuations and differences in the path delays. Also, the time difference between two sensor nodes may become large over time due to the wandering effect of the local clocks.

A proposed solution called *post-facto synchronization* [21] aims to provide an "instant" of synchronized time. When a stimulus occurs in a sensor field, the neighbor nodes observing this stimulus record the time of the event with respect to their local clocks. Afterwards, a third party node acting as a beacon broadcasts a synchronization pulse. The neighbor nodes use the synchronization pulse as a reference and normalize their stimulus time-stamps with respect to the reference. The post-facto synchronization is not appropriate for security time-stamp and target tracking applications. It is also limited to the transmission range of the beacon. As a result, if the sensor field is large, the events occurring in different parts of the field may be disjointed in time.

Another proposed solution called *Reference-Broadcast Synchronization* (RBS) [22] also aims to provide instantaneous synchronization among a set of receivers that are within the reference broadcast of the transmitter, which broadcasts m reference packets. Each of the receivers within the broadcast range records the time-of-arrival of the reference packets. The receivers then communicate with each other to determine the offsets. To provide multi-hop synchronization, it is proposed to use the so-called translation nodes, which are receiving two or more reference broadcasts from different transmitters. These translation nodes are used to translate the time between different broadcast domains.

The RBS scheme [22] has been studied using one broadcast domain to achieve

microsecond order precision. It is argued that RBS can provide multi-hop synchronization by translating the time between different domains. However, the impact of the translation errors and delays on the synchronization still needs to be studied. In addition, the energy dissipation and effects of node mobility in large scale sensor networks, e.g., 300, 1000, and 2000 nodes, need to be addressed. Also, events occurring in different parts of the sensor field may have very different time-of-occurrences at different sinks in the sensor field. As a result, this may cause problems if the sinks need to coordinate with each other. Furthermore, the overhead of translation may be significant if RBS is used to provide time based medium access.

In Internet, the *Network Time Protocol* (NTP) [47] is used to discipline the frequency of each node's oscillator. It maybe useful to use NTP to discipline sensor nodes, but the sensor nodes may be off when power management and topology maintenance protocols, e.g., SPAN [13], GAF [76], and LEACH [33], are employed. In addition, disciplining all the sensor nodes in the sensor field may be a problem due to interferences from the environment and large variation of delays between different parts of the sensor field. The interferences can temporarily disjoint the sensor field into multiple smaller fields causing undisciplined clocks among these smaller fields.

A more recently developed *Time-Sync protocol for Sensor Networks* (TPSN) [26] is based on similar methodology as the NTP, where the sensor nodes are organized into multiple levels and synchronized to the root node of the hierarchy. The nodes at the first level synchronize to the root node. After the synchronization, the nodes at the second level synchronize to the first level. This synchronization process repeats until all nodes in the network are synchronized. Unlike the Internet, the root node and nodes at different levels responsible for synchronization may fail often. As a result, this situation may cause some of the nodes in the network to be unsynchronized. In addition, this scheme may experience problems when the sensor nodes are mobile. The sensor nodes may move out of their hierarchy levels, and such scenario may disrupt the

synchronization procedure that depends on predefined level-by-level synchronization.

To provide network-wide time synchronization, the time differences among the sensor nodes must be minimized before protocols requiring time-stamps, e.g., security applications, flow control protocols, target tracking, voice fusion, video fusion, and environmental data fusion, are realizable. In addition, the time synchronization protocol must be robust to node failures as well as energy consumption in the network. Also, node mobility must be taken into account.

As a result, the *Time-Diffusion Synchronization Protocol* (TDP) is designed to provide network-wide time synchronization. The motivations for network-wide time synchronization are as follows:

- *Enable applications to coordinate sensor nodes, e.g., target tracking, data fusion, and decision fusion.*
- *Enable users to perceive events in the same time frame, e.g., multiple fire outbreaks at different locations of the sensor field.*
- *Enable protocols that require time-stamps, e.g., security, flow control, and medium access protocols.*

The TDP is used to maintain the time throughout the network within a certain tolerance. The tolerance level can be adjusted based on the application of the sensor networks. One of the benefits of TDP is that the performance of voice and video applications can be improved when multiple sources are sending data back to the sink through flooding or *directed diffusion* [37]. The TDP enables the sink to detect the time difference between multiple sources, so that the temporal differences can be adjusted. In addition, it allows the sink to issue a start time to the sensor nodes allowing interactive sensing and monitoring.

The design issues and system architecture of TDP are described in Section 3.2.

The TDP protocol is presented in Section 3.3. The analytical performance and simulation results are discussed in Sections 3.4 and 3.5, respectively.

3.2 Design Issues and System Architecture

Small and low-end sensor nodes may exhibit device behaviors that may be much worse than the large systems such as *personal computers* (*PCs*). As a result, the time synchronization in these nodes presents a new challenging problem. Some of the factors influencing time synchronization in large systems also apply to sensor networks [43]; they are *temperature*, *phase noise*, *frequency noise*, *asymmetric delays*, and *clock glitches*.

- **Temperature:** Since sensor nodes are deployed in various places, the temperature variations throughout the day may cause the clock to speed up or slow down. For a typical PC, the clock drifts few *parts per million* (*ppm*) during the day [48]. For low-end sensor nodes, the drifting may be even worse.

- **Phase noise:** Some of the causes of phase noise are access fluctuations at the hardware interface, response variation of the operating system to interrupts, and jitter in the network delay. The jitter in the network delay may be due to medium access and queueing delays.

- **Frequency noise:** The frequency noise is due to the unstability of the clock crystal. A low-end crystal may experience large frequency fluctuation, because the frequency spectrum of the crystal has large sidebands on adjacent frequencies.

- **Asymmetric delay:** Since sensor nodes communicate with each other through the wireless medium, the delay of the path from one node to another may be different than the return path. As a result, an asymmetric delay may cause an offset to the clock that can not be detected by a variance type method [43]. If the asymmetric delay is static, the time offset between any two nodes is also static.

The asymmetric delay is bounded by one-half the round trip time between the two nodes [43].

- **Clock glitches:** Clock glitches are sudden jumps in time. This may be caused by hardware or software anomalies such as frequency and time steps.

Besides dealing with these factors, a time synchronization protocol for sensor networks should be *automatically self-configured* and be *sensitive to energy requirement*. These are the two design criteria that the TDP is engineered around. The TDP self-configures and self-organizes to address the frequent network partitioning caused by sensor node failures. In addition, the TDP does not depend on any particular node to be a time server/master node. Since the sensor nodes may be left unattended with portable batteries for a long period of time, the workload for time synchronization should be spread to all sensor nodes to prolong the lifetime of the network.

Unlike time synchronization in the Internet using NTP, the nodes in the sensor network can not depend on specialized time servers because of the above two design criteria. Although precise time servers are great for timing references, they may die at a much earlier time than the other nodes, which may result in an unsynchronized network eventually. If precise time servers are used, they should have significant long life-time batteries or dedicated power sources, e.g., AC sources. In the following paragraphs, the TDP is described for both cases: (i) *with precise time servers* and (ii) *without precise time servers*.

(i) *With precise time servers:* The overall system architecture of how TDP interacts with the outside world is shown in Figure 18. *The main objective of the TDP is to enable the time of the sensor nodes to reach an equilibrium time.* The *sinks* may act as precise time servers for the sensor nodes residing in the sensor field. They broadcast a reference time to all the master nodes in the sensor network; master nodes are sensor nodes randomly elected to synchronize their neighbors.

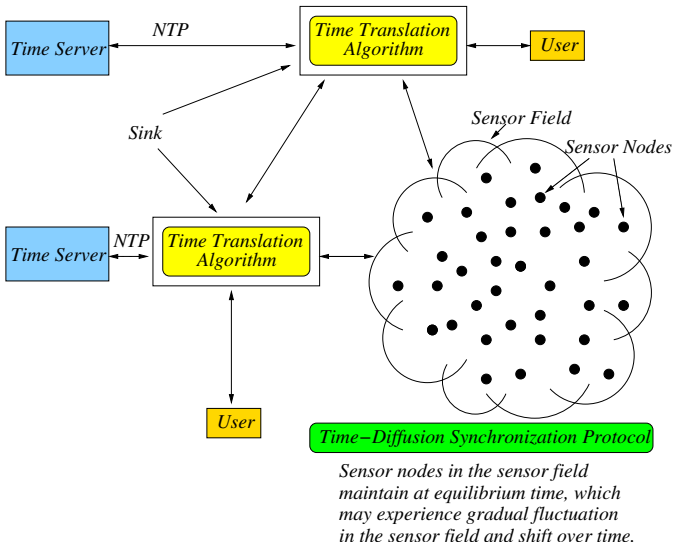

Figure 18: System architecture.

In turn, the master nodes use the received reference time to synchronize their neighbor nodes by using the TDP. In essence, the equilibrium time that the sensor network reaches is the reference time broadcast by the sinks.

(ii) *Without precise time servers:* Although the TDP can be used with precise time servers, it is more important to discuss about the autonomous nature of TDP since the line-of-sight or connection to all master nodes from the sinks may not be possible. Also, the sensor network may be deployed in areas that may not be accessed by the sinks for a long period of time, e.g., caves and ocean floor. Consequently, the sinks may not be used as time servers; fortunately, the autonomous nature of the TDP allows the sensor network to reach an equilibrium time that is independent from the time used by the Internet, e.g., *Universal Coordinated Time* (*UTC*).

Since the time in the sensor network reaches an equilibrium value, it still may drift over time and has fluctuation throughout the sensor network. *From the perspective of the outside world, the sensor network is like a multi-dimensional clock, where the time varies in space, i.e., sensor field, and time.* Although the time variation throughout

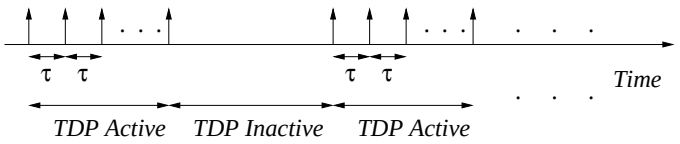

Figure 19: TDP active/inactive schedule.

the sensor network may be very small, it is necessary to translate the time in the sensor network to a common time, e.g., UTC, used by the users.

The sinks as shown in Figure 18 take care of such translation and serve as interfaces to the sensor network. They are synchronized to the time servers using the NTP [52]. The accuracy of the NTP synchronization is in the order of milliseconds [35]. The sinks use the UTC time as a reference and translate the time in the sensor network using the *Time Translation Algorithm* [71], which smoothes out the time variation by using a playout buffer. Consequently, the time presented to the users is almost the same. This type of technique is similar to *Jitter Time Stamp* [69]. Note that the focus of this particular paper is on the synchronization problem in the sensor field without precise time servers.

The time schedule for applying TDP is illustrated in Figure 19. During the active period, the master nodes are reelected at every τ seconds, which is a design parameter that depends on the types of sensor networks. The master nodes broadcast the timing information to their neighbors, which use this time as timing reference. The neighbor nodes self-determine to become diffused leader nodes that further broadcast the timing information to their neighbors. The duration of the TDP active period depends on the range of time variation allowed throughout the sensor network. On the other hand, the inactive period depends on the amount of clock drifts allowed before TDP is activated again. For example, the TDP is applied until the time within the sensor network has a difference of 100 milliseconds. It is inactive for a period of time that allows the clocks to drift apart by 50 milliseconds. When the time difference within the sensor network is 150 milliseconds, the TDP protocol is applied again. The

active and inactive periods are design parameters that can be tailored for different types of sensor networks. An overview of the procedures and functionality of the TDP is described in the following paragraphs.

The TDP architecture consists of many algorithms and procedures as illustrated in Figure 20. The TDP protocol focuses on all the algorithms and procedures except the *clock discipline algorithm*. The clock discipline algorithm may use the adaptive hybrid clock discipline algorithm intended for NTP Version 4 [48]. The hybrid clock discipline algorithm uses a combination of phase lock loop (PLL) and frequency lock loop (FLL), which are usually implemented in hardware to minimize the noise. For low-end sensor nodes, it may not be possible to have a combination of PLL and FLL due to monetary cost of each node. As a result, there may still be room for a different type of clock discipline algorithm specifically designed for low-end sensor nodes.

The algorithms and procedures in Figure 20 are used to autonomously synchronize the nodes, remove the false tickers (clocks that deviate from their neighbors), and balance the load required for time synchronization among the sensor nodes. Initially, the sensor nodes may receive an *Initialize pulse* from the sink either through direct broadcast or multi-hop flooding. Then they self-determine to become master nodes with the *election/reelection of master/diffused leader node procedure* (ERP), which consists of the *false ticker isolation algorithm* (FIA) and *load distribution algorithm* (LDA) as shown in Figure 20. At the end of procedure ERP, the elected master nodes start the *peer evaluation procedure* (PEP) while others do nothing. The procedure PEP helps to remove false tickers from becoming neither a master node nor a diffused leader node.

After procedure PEP, the elected master nodes (denoted by W in Figure 20) start the *time diffusion procedure* (TP), where they diffuse the timing information messages at every δ seconds (round interval) for a duration of τ seconds. Each neighbor node

(e.g., node B or C in Figure 20) receiving these timing information messages self-determines to become a diffused leader node using the procedure ERP. Furthermore, all neighbor nodes adjust their local clocks using *time adjustment algorithm* (TAA) and *clock discipline algorithm* (CDA) after waiting for δ seconds as shown in Figure 20.

The elected diffused leader nodes (e.g., node B) will further diffuse the timing information messages to their neighboring nodes (e.g., nodes D and E) within their broadcast range. Note that these timing information messages are diffused by each elected diffused leader node for n hops from the master nodes, where each hop represents one level from the master nodes (e.g., nodes B and C are at Level-1 while nodes D and E are at Level-2). This diffusion process enables all nodes to be autonomously synchronized. In addition, the master nodes are re-elected at every τ seconds using the procedure ERP, which is repeated for $\theta - 1$ times, where $\theta\tau$ is equal to the length of the TDP active period.

The functionality and novelties of the procedures PEP, TP, and ERP as shown in Figure 20 are described in the following section. The procedures PEP and TP are explained before the procedure ERP, because the algorithms FIA and LDA of procedure ERP require the understanding of these procedures.

3.3 Time-Diffusion Synchronization Protocol (TDP)

The TDP is composed of components as illustrated in Figure 20.

3.3.1 Peer Evaluation Procedure (PEP)

The purpose of procedure PEP is to allow neighbor nodes to evaluate the stability of the local clock by using the Allan variance [3]. The Allan variance is used to estimate the deviations between two clocks. The steps are as follows:

Step 1: Initially, the elected master nodes broadcast η number of time-stamped $SCAN$

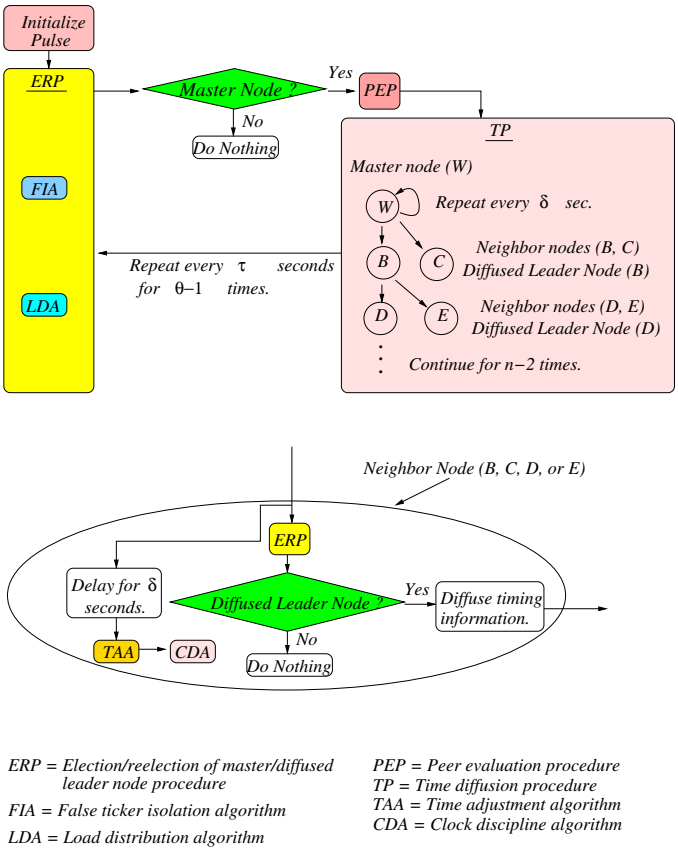

Figure 20: TDP architecture.

messages to the neighbor nodes within a short time interval, so the phase and frequency noises are almost white [43].

Step 2: The neighbor nodes use these $SCAN$ messages to calculate the 2-sample Allan variance $\sigma^2(\iota)$ [3, 48] of the local clock from the clock of the master nodes as follows:

$$\sigma^2(\iota) = \frac{1}{2\iota^2(N-2)} \sum_{g=1}^{N-2} (x_{g+2} - 2x_{g+1} + x_g)^2 \qquad (1)$$

where ι is the time difference between two time deviation measurements; N is the total number of time deviation measurements, and x is the measurement value.

Step 3: The calculated Allan variances $\sigma^2(\iota)$ by equation (1) are sent back to the master nodes with the $REPLY$ messages.

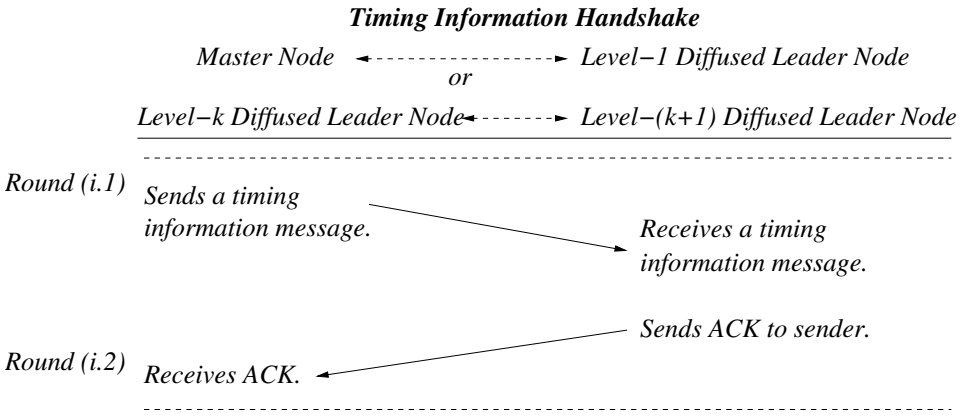

Figure 21: The handshake of timing information message.

Step 4: The master nodes then calculate the outlier ratio γ_{yz}, which indicates the amount of clock deviation, between the sensor node y (e.g., master node) and z (e.g., neighbor node) by

$$\gamma_{yz} = \mid \frac{\sigma^2_{yz}(\iota) - \sigma^2_{avg}(\iota)}{\sigma^2_{avg}(\iota)} \mid \tag{2}$$

and the average $\sigma^2_{avg}(\iota)$ of the Allan variances by

$$\sigma^2_{avg}(\iota) = \frac{\sum_{z=1}^{M} \sigma^2_{yz}(\iota)}{M} \tag{3}$$

where $\sigma^2_{yz}(\iota)$ is the Allan variance between nodes y and z calculated by equation (1); and M is the number of Allan variances received. In addition, the average $\sigma_{avg}(\iota)$ of the Allan deviations ($\sqrt{\sigma^2(\iota)}$) is determined as

$$\sigma_{avg}(\iota) = \frac{\sum_{z=1}^{M} \sqrt{\sigma^2_{yz}(\iota)}}{M} \tag{4}$$

As a result, the outlier ratio γ_{yz} (equation (2)) and the average Allan deviation $\sigma_{avg}(\iota)$ (equation (4)) are sent back to the neighbor nodes using the $RESULT$ messages.

Step 5: Repeat steps 1 to 4 for n hops from the master nodes, where in each hop the elected diffused leader nodes are the ones that broadcast η number of time-stamped $SCAN$ messages and perform the evaluation of their neighbor nodes.

Step 6: At the end of τ period, reset the Allan variances (equation (1)) and outlier ratios (equation (2)) to zero and start the same procedure with them from the master nodes.

After the procedure PEP, all sensor nodes receive the outlier ratios γ_{yz} (equation (2)) and the average Allan deviation $\sigma_{avg}(\iota)$ (equation (4)), which are used to evaluate the quality of their clocks with respect to their neighbors by the procedure ERP. In the following section, the procedure TP is described.

3.3.2 Time Diffusion Procedure (TP)

As shown in Figure 20, the procedure TP diffuses timing information messages from the master nodes to the neighboring nodes, where the timing information messages are further diffused by the elected diffused leader nodes for n hops from the master nodes. At the end of this procedure, the timing information is used by the algorithm TAA to adjust the local clocks.

3.3.2.1 Timing Information Handshake

The timing information messages are diffused from one level to the next starting from the master nodes, and it contains the following fields:

1. *Master node local ID (M-LID)* (ID of the master node of which it is originated),
2. *Source LID* (the LID of the node that broadcasts the timing information message),
3. *Value n* (the number of levels that the timing information message is to be diffused),

4. *Time* $t_{M,i}$ (the diffused time from the master LID M that neighbors should synchronize to at Round(i)), and

5. *Value* $\beta_{M,k}$ (the value used by the algorithm TAA to calculate the weight for the diffused time $t_{M,i}$ at Level-k).

The elected diffused leader nodes at Level-1 respond to the timing information messages with ACK messages (Round (i) in Figure 21). Afterwards, the round trip time Δ_j between the master node and diffused leader node j is calculated by

$$\Delta_j = (t_1 - t_0) \tag{5}$$

where t_1 is the arrival time of the ACK message and t_0 is the broadcast time of the timing information message at Round (i) in Figure 21.

Since each master node may receive multiple ACK messages, the average Δ of the round trip delays Δ_j (equation (5)) is calculated and used to estimate the one-way delay between the master node and the neighboring nodes. As a result, the diffused time $t_{M,i}$ from the master nodes can be calculated as

$$t_{M,i} = t_{M,i} + \Delta/2 + \delta \tag{6}$$

where $\Delta/2$ is the estimated one-way delay, and δ is the amount of time that the neighboring nodes wait before adjusting their local clocks.

Furthermore, the standard deviation α of the round trip delays Δ_j (equation (5)) is obtained and used to estimate the quality of the diffused time $t_{M,i}$. Note that a large α value means that the diffused time $t_{M,i}$ may have a larger error. Hence, the standard deviation α is accumulated at every hop starting from the master node. This accumulated deviation value $\beta_{M,k}$ is calculated as

$$\beta_{M,k} = \beta_{M,k-1} + \alpha \tag{7}$$

where k is the distance from the master node in terms of the number of hops.

The elected diffused leader nodes follow the same handshake procedure when they propagate the timing information message from Level-k to Level-(k+1) as shown in Figure 21 with the following modifications:

- The time $t_{M,i}$ is calculated and only adjusted by $\Delta/2$ as

$$t_{M,i} = t_{M,i} + \Delta/2 \tag{8}$$

- The value n, the number of levels to be diffused, is decreased by one after each broadcast. A diffused leader node will not propagate the timing information message if the value n stored in the received timing information message is set equal to 0.

- The source LID in the timing information message is set to the LID of the diffused leader node.

- The value $\beta_{M,k}$ is calculated by equation (7).

- The M-LID value stays the same in the timing information message.

Moreover, the master nodes diffuse a new timing information message every δ seconds (i.e., time between two rounds) for a duration of τ seconds to address the clock wandering and mobility of nodes. In addition, the timing information message handshake is repeated level-by-level by the elected diffused leader nodes as shown in Figure 21.

Furthermore, a sensor node may receive multiple timing information messages from different master nodes. The information in these timing information messages are stored in a *timing data table* and used later by the algorithm TAA to calculate the new time for the node.

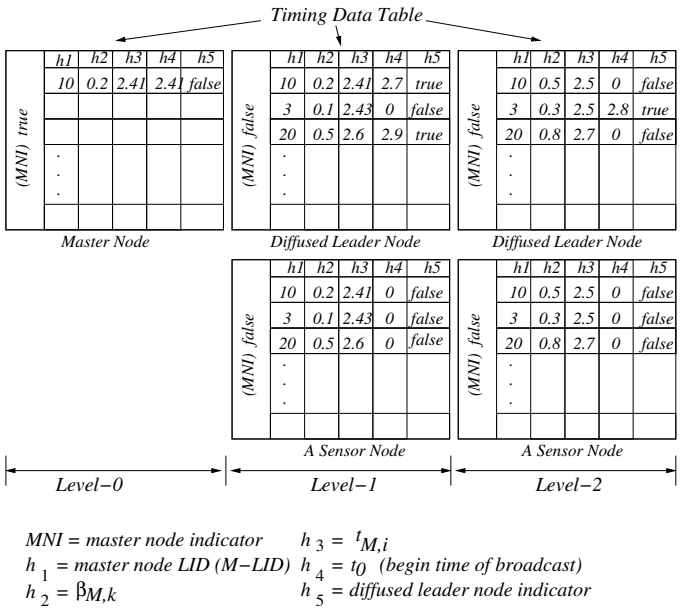

Figure 22: An example of timing information stored in the nodes.

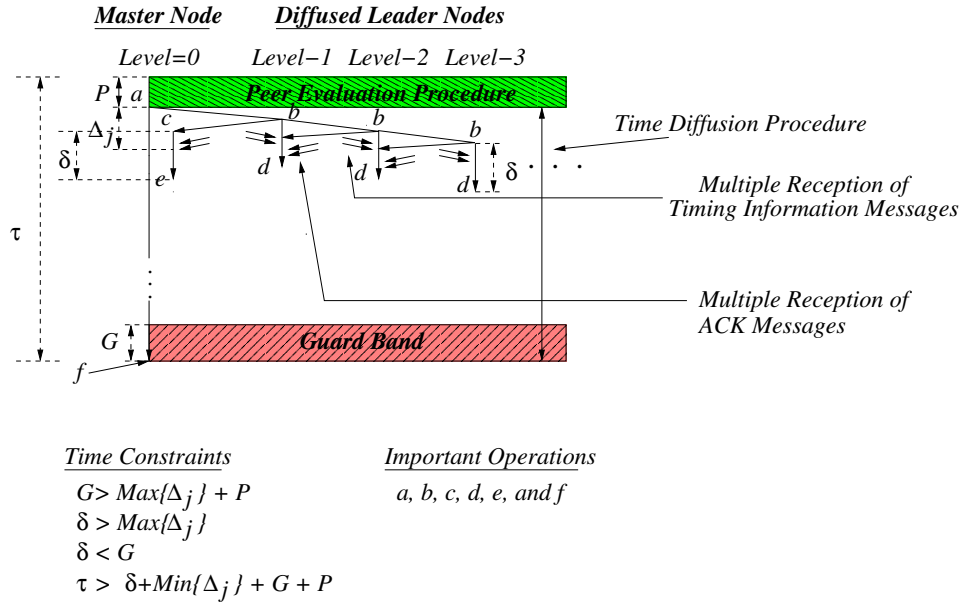

Figure 23: The timing diagram of the timing information handshake.

An example of the timing information stored in the timing data table is illustrated in Figure 22. It shows the contents of a master node, Level-1 neighboring nodes, and Level-2 neighboring nodes. The details of the contents are described as follows:

Level-0: A master node sets the *master node indicator* (MNI) to true in the timing data

table. Each row of the table contains the information related to the timing information message diffused by the master node. For example, the M-LID, $\beta_{M,k}$, $t_{M,i}$, and t_0 values are stored under columns h_1 to h_4 (e.g., M-LID =10; $\beta_{M,k} = 0.2$; $t_{M,i} = 2.41$; and $t_0 = 2.41$). In addition, column h_5 is set to false since a master node can not be elected as a diffused leader node. Also, a master node should only have one row of data in the table, because it is diffusing its time.

Level-1: There may be many nodes, which are one hop away from the master nodes. Some of them may be elected diffused leader nodes while others are not. The contents of both categories of nodes are shown in Figure 22. In addition, the MNI of both categories is set to false indicating that the nodes are not master nodes. The example illustrated in Figure 22 shows that both categories of nodes receive three timing information messages from three master nodes, e.g., M-LID = 10; M-LID = 3; and M-LID = 20. The only differences between the contents of both categories are under columns h_4 and h_5. For each received timing information message, a node may be elected as a diffused leader node. If it is elected, it is specified in column h_5 with the value true (e.g., true for M-LID = 10 and M-LID = 20 while false for M-LID = 3 of the elected diffused leader node). If a node is not elected, column h_5 is specified with the value false and h_4 is set to 0 since the node is not diffusing any timing information message. As a result, a diffused leader node may have both false and true indicator values while a node elected not to diffuse any timing information message has only false values for all rows in column h_5.

Level-2: The method of storing information in the timing data table is the same as Level-1. Note that the values under columns h_2 and h_3 are different than the values in Level-1, because they are adjusted by equations (7) and (8). In addition,

the diffused leader node indicators at column h_5 are different, because a sensor node at Level-2 may be elected differently than a node in Level-1 to diffuse the timing information message, which originates from the same master node. For example, a diffused leader node at Level-2 is elected to diffuse the timing information message from M-LID = 3 at time 2.8 sec while the diffused leader node at Level-1 is not elected although both of these nodes receive the timing information message from M-LID = 3.

Since the timing information message handshake involves message exchanges as shown in Figure 21, it is important to understand the time constraints between these message exchanges. As a result, the timing diagram of the timing information message handshake is described in the following section.

3.3.2.2 Timing Diagram of the Timing Information Handshake

The TDP consists of three procedures ERP, PEP, and TP as shown in Figure 20. Since the procedure ERP requires a small amount of processing time comparing to procedures PEP and TP, it is not included in the timing diagram as shown in Figure 23, which captures the timing relationship of events within τ period. The procedure PEP given in Section 3.3.1 requires P seconds of processing time while the procedure TP occupies the rest of the τ period. In the following paragraphs, the timing relationship of the procedure TP is described.

As shown in Figure 23, the procedure TP consists of the operations a, b, c, d, e, and f as well as the *guard band*. The time constraints of the operations and *guard band* are described as follows:

- *Guard band:* G seconds long, where $G > Max\{\Delta_j\} + P$; it prevents operations a, b, c, and e from occurring and generating events that may spill over to the next τ period. The operations are initiated when an ACK message is received at Δ_j (equation (5)) seconds after operation a.

- $b \rightarrow d$ *and* $c \rightarrow e$*:* δ seconds long, where $\delta > Max\{\Delta_j\}$ and $\delta < G$. This is to ensure that all timing information and ACK messages are received within δ seconds when diffusing a timing information message.

- τ *period:* $\tau > \delta + Min\{\Delta_j\} + G + P$; it ensures that a timing information message is diffused for at least one round.

Within the τ period, the operations a through f are performed during each round, which is kept tracked by the θ counter, where $\theta\tau$ is the TDP active period. The functionality of each operation is as follows:

Operation a: The master nodes broadcast timing information messages.

Operation b: The sensor nodes perform the procedure ERP detailed in Section 3.3.3, which elects the diffused leader nodes as described in Section 3.3.3. The elected diffused leader nodes send an ACK message and broadcast a timing information message as shown in Figure 23. In addition, they initiate an operation d occurring δ seconds later. Before operation d takes place, the diffused leader nodes may receive multiple ACK messages. Every time an ACK message is received, the round trip delay Δ_j is measured using equation (5). In addition, the average Δ and standard deviation α of the round trip delays Δ_j are calculated and stored in the timing data table as described in Section 3.3.2.1.

Operation c: The master nodes receive an ACK message from Level-1 diffused leader nodes and initiate operation e occurring δ seconds later as shown in Figure 23. Before executing operation e, multiple ACK messages may be received. As a result, the average Δ and standard deviation α of the round trip delays Δ_j are calculated.

Operation d: The sensor nodes adjust their local clocks with the algorithm TAA. They also remove all rows in the timing data table. In addition, the Δ_j values are cleared

while the α and Δ values are kept, which are used to calculate $\beta_{M,k}$ and $t_{M,i}$ with equations (7) and (8), respectively.

Operation e: The master nodes clear the timing data table and initiate operation a.

Operation f: All sensor nodes in the sensor network reset their variables. For instances, the master node indicator (*MNI*) is set to true, and all the rows in the timing data table are cleared. In addition, the Δ_j values are cleared while the Δ and α values are kept. Furthermore, the θ counter is decreased by one, and the master nodes broadcast a *SYNCH* message containing the value of the θ counter. The *SYNCH* message is intended for new sensor nodes that have been just added into the network. Once these new sensor nodes receive the *SYNCH* message, they set the θ counter to the value specified in the *SYNCH* message. Only new sensor nodes that have received the *SYNCH* message can participate in becoming a master or diffused leader node with the procedure ERP. The rest of the new sensor nodes has to wait until it has received the *SYNCH* message that only occurs at every τ seconds. Since the operation f is at the end of the τ period, the procedure ERP is performed, which elects new master nodes for the next τ period.

The above operations are carried out during each round of timing information message diffusion and continue until until the θ counter reaches 0. In the following section, the procedure ERP is described specifying how a master or diffused leader node is elected.

3.3.3 Election/Reelection of Master/Diffused Leader Node Procedure (ERP)

As shown in Figure 23, the diffused leader and master nodes are elected in operations b and f, respectively. Both types of elections depend on the outputs of the algorithms FIA and LDA to automatically self-configure the nodes as described in Sections 3.3.3.1 and 3.3.3.2.

3.3.3.1 False Ticker Isolation Algorithm (FIA)

The algorithm FIA uses the outlier ratio outputs (equation (2)) of the procedure PEP as described in Section 3.3.1 to self-determine if a node is a false ticker or not. If the average of these received outlier ratios is greater than the threshold ϕ, the sensor node is an outlier. The value ϕ controls the quality of the selected clocks. For instance, a small ϕ value means that the selected clocks have small deviations with the clocks of the neighbor nodes. When the average outlier ratio is greater than 1, it means that the local clock deviates from the clocks of the neighbor nodes by more than twice the average Allan variance given by equation (3).

If a node self-determines to be an outlier, it is a false ticker. The false ticker does not become neither a diffused leader node during the current τ period nor a master node at the beginning of the next τ period. The algorithm FIA aims to remove nodes that have high frequency noise clocks or high access fluctuation due to either network jitter or access variations from becoming master or diffused leader nodes. If a node is not a false ticker, then it uses the algorithm LDA as described in Section 3.3.3.2 to determine if it is elected as a master or diffuse leader node.

3.3.3.2 Load Distribution Algorithm (LDA)

Besides allowing the sensor network to achieve an equilibrium time, the TDP needs to be energy efficient and capable of distributing the energy consumption for diffusing time to all sensor nodes in the network. It achieves them by reelecting master and diffused leader nodes at every τ and δ seconds, respectively. During the reelection, the nodes randomly choose a value λ that is between 0 and 1. The value λ is then shifted by the value $(1 - \zeta)$, where ζ is the ratio of current energy level over the maximum allowed energy level, and calculated as

$$\lambda = \lambda - (1 - \zeta) \tag{9}$$

If the value λ is greater than the threshold φ, then the node is either a master or diffused leader node depending if the master or diffused leader node is being reelected. The threshold φ determines the number of sensor nodes participating as a master or diffused leader node. For example, if φ is set equal to 0.7, it means on the average that 30 percent of the deployed sensor nodes is a master node or diffused leader node. As a result, $\rho = 1-\varphi$ represents the fraction of deployed sensor nodes that is a master or diffused leader node. For this case, ρ is set equal to 0.3.

Since the shifting of the randomly selected value λ is based on the current energy level of the sensor node, ρ decreases if the threshold φ is not adjusted appropriately. As a result, the threshold φ stored in all sensor nodes is adjusted at every τ seconds according to

$$\varphi = \varphi - \varepsilon \tag{10}$$

where ε is the amount that needs to be adjusted, which is based on μ (energy consumed per round of timing information message diffusion). The value μ can be approximated by

$$\mu \approx \frac{Amount\ of\ energy\ consumed\ during\ \tau\ seconds}{\lceil \tau/\delta \rceil - 1} \tag{11}$$

where τ is the master node reelection period, and δ is the time between each round of timing information message diffusion.

As a result, the value ε (see Appendix A) is calculated as

$$\varepsilon \;=\; \rho - \sum_{m=1}^{i} \Phi_{i,m} \rho^{m-1} (1-\rho)^{(i-m)} (\rho - (m-1)\epsilon) \tag{12}$$

where ρ is the fraction of sensor nodes that can become a master or diffused leader node; i is the number of rounds within a τ period, which is approximated by $\lceil \tau/\delta \rceil - 1$;

ϵ is the ratio of μ (equation (11)) over the maximum energy level; and the coefficient $\Phi_{i,m}$ is calculated as

$$\Phi_{i,m} = \begin{cases} 1 & , \ for \ m = 1 \\ \sum_{j=1}^{i-1} 1 & , \ for \ m = 2 \\ (\sum_{v_{m-1}=1}^{i-(m-1)} (\sum_{v_{m-2}=1}^{i-(m-2)-v_{m-1}} (... & , \ for \ m \geq 3 \\ (\sum_{v_1=1}^{i-1-\sum_{k=1}^{m-2} v_{m-k}} 1)...))) & \end{cases} \tag{13}$$

with $i \geq m$ and $m-1$ levels of summation for $m \geq 3$, e.g., $(\sum(\sum \cdot))$ and $(\sum(\sum(\sum \cdot)))$ are 2 and 3 levels, respectively.

3.3.4 Time Adjustment Algorithm (TAA)

As shown in Figure 21, a sensor node at operation d adjusts its local clock with time $t_{M,i}$ and deviation $\beta_{M,k}$, which are obtained from columns h_2 and h_3 of its timing data table. First, the node sums up all the deviations in column h_2 of the timing data table, where β_T denotes the sum and set $\mathbf{S}$ contains all the deviations $\beta_{M,k}$. In addition, set $\mathbf{T}$ contains all the times $t_{M,i}$ in column h_3 of the timing data table, where $\mid \mathbf{T} \mid = \mid \mathbf{S} \mid$.

Let u be the u^{th} element stored in sets $\mathbf{S}$ and $\mathbf{T}$, where both u^{th} elements in sets $\mathbf{S}$ and $\mathbf{T}$ are obtained from the same timing information message, which occupy the same row in the timing data table. For example, S_u and T_u are the u^{th} elements in sets $\mathbf{S}$ and $\mathbf{T}$, respectively.

As a result, the weight ω_u for the diffused time T_u is determined as

$$\omega_u = \xi_u / \xi_T \tag{14}$$

where ξ_u (the unnormalized weight for the diffused time T_u) is calculated as

$$\xi_u = \beta_T - \mathrm{S}_u \tag{15}$$

and ξ_T (the normalizing factor for the weight ξ_u) is determined as

$$\xi_T = \sum_{u=1}^{|\mathbf{S}|} \xi_u \tag{16}$$

The weight ω_u is large when the deviation S_u is small. As a result, the sensor node uses more of the time diffused by a master node that is a hop away than two hops away. In addition, the effect of the asymmetric delay is lowered since the asymmetric delay is bounded by one-half of the round trip delay [43], and the round trip delay of one hop should be rather small, i.e., milliseconds order.

Once the weight ω_u for each diffused time T_u is obtained from equation (14), it is used to calculate the new time t_{new} for the node. If the set $\mathbf{T}$ is empty, then the new time is just the current local time (t_{local}), without any change. If the set has only one element T_1, then the new time t_{new} is set equal to that element. Otherwise, all the elements in set $\mathbf{T}$ are weighted by ω (equation (14)), and they are summed up to provide the new time t_{new}. In summary, t_{new} is calculated as

$$t_{new} = \begin{cases} t_{local} & , \ for \ \mid \mathbf{T} \mid = 0 \\ \mathrm{T}_1 & , \ for \ \mid \mathbf{T} \mid = 1 \\ \sum_{u=1}^{|\mathbf{T}|} \omega_u \cdot \mathrm{T}_u & , \ for \ \mid \mathbf{T} \mid \geq 2 \end{cases} \tag{17}$$

As the new time t_{new} is calculated by equation (17), the local clock is not updated with the new time if $\mid t_{local} - t_{new} \mid$ is smaller than the average of the received Allan deviations, which are the outputs of procedure PEP as described in Section 3.3.1. This is to prevent unnecessary updates to the local clock since the new time t_{new} is within the range of clock deviation among the neighbors.The TDP is composed of components as illustrated in Figure 20.

3.4 Analytical Performance Evaluation

It is important to show that TDP can allow the time in the sensor nodes to converge to an equilibrium time with a small variation that is equal to the round trip time υ between two adjacent neighboring nodes. The value υ consists of two components: (1) the processing delays and (2) the propagation delays. Since the propagation delays may be in the order of microseconds, the time precision between nodes is gated by the processing delays. As a result, υ can be controlled by varying the processing delays, which consist of the medium access and queueing delays.

Each node is assumed to have received at least 2 timing information messages. The minimum ρ value (ρ_{min}) required to satisfy this assumption is calculated as

$$\rho_{min} = \frac{l \cdot w}{\kappa\pi(R \cdot n)^2} \tag{18}$$

where l and w are the length and width of the sensor field, respectively; κ is the number of nodes deployed in the sensor field; R is the broadcast radius of a sensor node; and n is the number of levels that the timing information message is to be diffused.

Although ρ_{min} gives the minimum value of ρ, ρ is best to be few times larger than ρ_{min}. This is to account for uneven distribution of nodes in the sensor field. By requiring the nodes to receive at least 2 timing information messages, the time in the sensor nodes can be diffused more effectively by using TDP.

The time deviation from the ideal time is assumed to be uniformly distributed between lower bound ϑ_{lower} and upper bound ϑ_{upper}. The range between ϑ_{lower} and ϑ_{upper} is separated into discrete sections with step size of υ. As a result, each section has the probability of $1/\lceil(\vartheta_{upper} - \vartheta_{lower})/\upsilon\rceil$ as shown in Figure 24. Since after each round of diffusion from the master nodes, the range between ϑ_{lower} and ϑ_{upper} shrinks. For the analysis, it is assumed that the range is shrunk by $\upsilon \cdot \nu_i$ seconds from the upper bound ϑ_{upper} and lower bound ϑ_{lower}, where ν_i is the shrink in multiples of υ

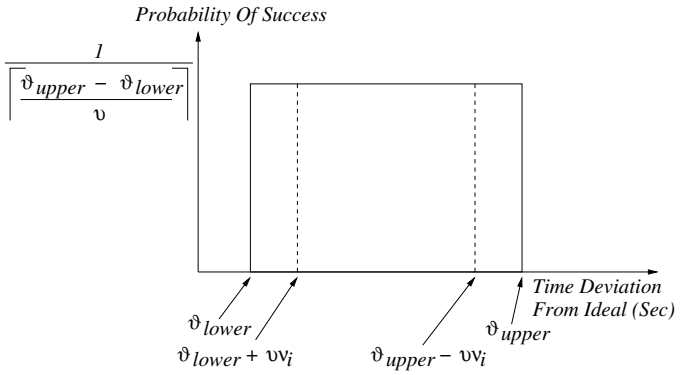

Figure 24: The probability distribution of the deviated time from the ideal.

at Round (i). After the range is shrunk, the ϑ_{upper} and ϑ_{lower} values are set to the new shrunk upper and lower bound values.

The probabilities $Q_{L,i}$ and $Q_{U,i}$ are the probabilities of shrinking by $\upsilon \cdot \nu_i$ from the lower and upper bounds at Round (i), respectively. Since the upper and lower bounds are assumed to shrink the same range, $Q_{L,i}$ is equal to $Q_{U,i}$ and is calculated as

$$Q_{L,i} = Q_{U,i} = \frac{\nu_i}{\lceil (\vartheta_{upper} - \vartheta_{lower})/\upsilon \rceil} \tag{19}$$

The objective of each round of timing information diffusion is to shrink the range of the deviated time distributed throughout the sensor network. The amount of shrinkage at Round(i) is $\upsilon \cdot \nu_i$. As a result, the probability Ω_i that all sensor nodes are shrunk by $\upsilon \cdot \nu_i$ at Round(i) is calculated as

$$\begin{aligned}
\Omega_i &= \Omega(\vartheta_{lower} + \upsilon \cdot \nu_i) < t < (\vartheta_{upper} - \upsilon \cdot \nu_i), \\
&= \binom{\kappa}{\lceil \rho\kappa \rceil} (1 - (Q_{L,i} + Q_{U,i}))^{\rho\kappa} (Q_{L,i} + Q_{U,i})^{(1-\rho)\kappa}, \\
&= \binom{\kappa}{\lceil (1-\rho)\kappa \rceil} (Q_{L,i} + Q_{U,i})^{(1-\rho)\kappa} \cdot \\
&\quad (1 - (Q_{L,i} + Q_{U,i}))^{\rho\kappa},
\end{aligned}$$

where κ is the number of nodes deployed in the sensor field; ρ is the fraction of

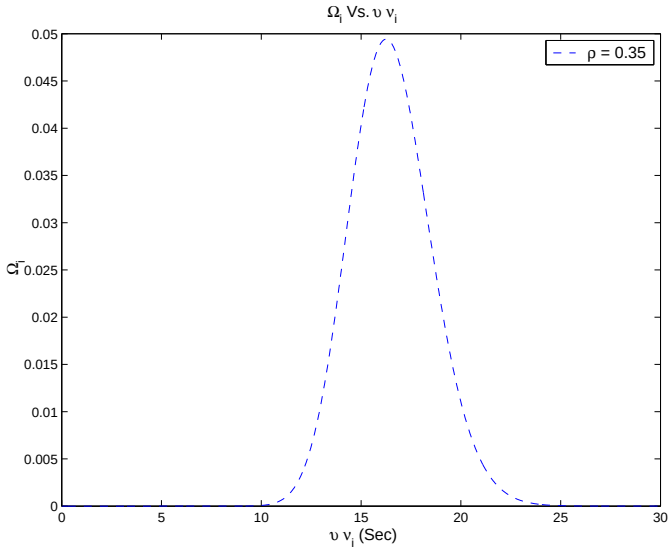

Figure 25: Ω_i vs. $\upsilon\nu_i$.

nodes that will become master nodes or diffused leader nodes; and $Q_{L,i}$ and $Q_{U,i}$ are probabilities calculated by equation (19). Furthermore, Ω_i can be approximated by the Poisson distribution if κ is large and Λ is small,

$$\Omega_i \simeq \frac{\Lambda^{\lceil(1-\rho)\kappa\rceil}}{\lceil(1-\rho)\kappa\rceil!}e^{-\Lambda} \quad , where\ \Lambda = Q_{L,i} + Q_{U,i}. \tag{20}$$

To show that the time distributed in the sensor network will converge to an equilibrium time, the probability given by equation (20) is plotted in Figure 25 for $\rho = 0.35$, $\kappa = 100$, $\vartheta_{upper} = 60\ sec$, and $\vartheta_{lower} = 10\ sec$. The maximum value of Ω_i occurs at around $\upsilon\nu_i = 16$ seconds. From equation (20) and Figure 25, there is a shrink range $\upsilon\nu_i$ that will give a maximum value of Ω_i for each round of timing information diffusion. After each round, the upper and lower bounds of the deviated time, i.e., ϑ_{upper} and ϑ_{lower}, are decreased by the shrink range $\upsilon\nu_i$. Once the deviated time range is shrunk, the probability distribution is still assumed equally distributed for the new range, and equation (20) is used to find the next best shrink range $\upsilon\nu_i$ at Round (i). This process is repeated for every round until the deviated time range is 2υ or less.

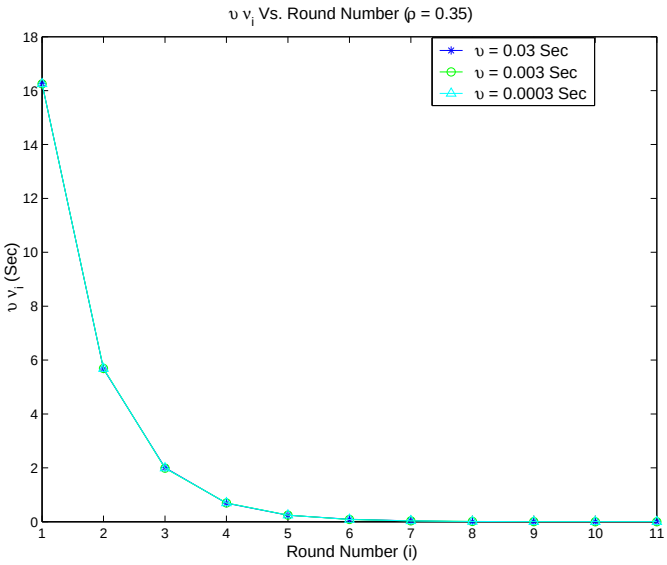

Figure 26: $\upsilon\nu_i$ vs. diffusion round.

The convergence of the proposed protocol based on this process is shown in Figure 26.

As shown in Figure 26, the shrink range decreases exponentially as the number of rounds of timing information message diffusion increases. This means that the range of deviated time throughout the network is slowly reaching its equilibrium time. Note that the convergence does not depend on the υ value, i.e., the round trip time between nodes. This means that the time in the network can reach different level of precision, e.g., milliseconds or microseconds order. In reality, the attainable order of precision may only be in milliseconds, because the error budgets for processing and queueing delays in real systems are in the microseconds range. Also, the convergence exhibited by TDP only depends on the number of timing information message diffusion. As an example, the time takes 7 rounds to converge when $\upsilon = 0.03$ *sec* while 9 and 11 rounds when $\upsilon = 0.003$ *sec* and $\upsilon = 0.0003$ *sec*, respectively. Note that these number of rounds are obtained based on the best probability Ω_i at each Round (i). In addition, the analysis is based on $\kappa = 100$, i.e., the number of nodes being deployed, but it is also valid for higher values of κ.

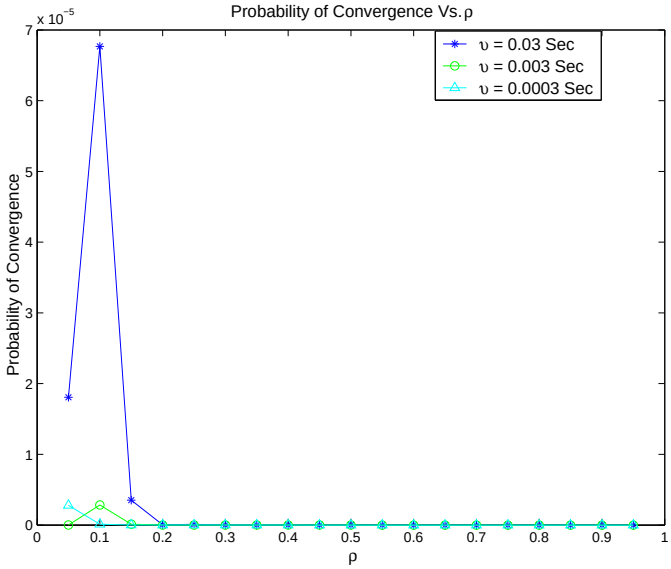

Figure 27: Probability of convergence vs. ρ.

The choice of ρ in equation (20) is also important. For $\kappa = 100$, the best ρ value with the highest probability of convergence is around 0.1 as shown in Figure 27. The probability of convergence is calculated as the product of Ω_i for $i = 1$ to (i when $\upsilon\nu_i$ is equal to 2υ or less). As κ increases to 200, the best ρ value shifts to around 0.3. In addition, the ρ value should be less than 0.5. Hence, half of the deployed nodes can be adjusted by the TDP since master nodes do not adjust their time although they receive the timing information messages from other master nodes.

3.5 Performance Evaluation By Simulation

The performance of the TDP is evaluated with an event driven simulation. Two hundred sensor nodes are deployed randomly in a 80 meters by 80 meters sensor field. Each of the sensor nodes can receive and transmit messages to its neighbors by executing the TDP independently, i.e., each sensor node is emulating a physical sensor node where it has its own memory. In addition, it keeps track of its own local time with a randomly selected drift rate that is between ± 100 ppm. Since each node keeps track of its local time, simulations with large number of nodes, e.g., 1000, 2000,

and 3000, may become difficult. It is because the simulation has to create an event for every clock tick. As a result, only 200 sensor nodes are deployed with the targeted precision of 10^{-1} seconds order. It is shown in Section 3.4 that TDP will work for higher order of precision. To show that TDP is able to reach its equilibrium time and maintain a small variation of the deviated time throughout the network, the local time of each sensor node is initially shifted by a random amount ranging from 10 seconds to 60 seconds from the ideal time. Since the local times of the neighbor nodes are quite different, this setup also shows how TDP recovers from network partitioning. In essence, this setup represents the worst case scenario in synchronizing time, where each node may be drifted far apart from each other.

When a node receives and transmits messages, it will consume power. It is assumed that a node does not go into the idle state while running the TDP since the nodes are active during the short period of time when TDP is running. All the nodes participate in TDP, and the timing information messages are diffused 3 hops from the master nodes for all simulations. The configuration of each node is listed in Table 8, which has the parameters as in [32] but with energy set to 1 Joule (J).

As specified in Table 8, the *processing delays* are 0.05 seconds and 0.01 seconds for saturated and not saturated sensor network, respectively. They are composed of delays incurred at the lower layers, i.e., medium access and physical layers. For example, a sensor node, which is equipped with an 802.11 MAC, may have a processing delay of around 0.01 seconds when the network is not saturated [44]; the delay value does not change for different node densities, e.g., 5, 10, and 20 nodes, and it is near constant until the network becomes saturated at around 75% of the channel capacity. At saturation, the processing delay of an outgoing message, the number of retransmission, and the end-to-end delay flatten at different values [44][72][18][7] depending on the 802.11 MAC parameters, e.g., node density and congestion window size. The processing delay for a saturated sensor network is assumed to be 0.05 seconds [44].

Table 8: Configuration of each sensor node.

Parameters	**Value**
Transmission radius	10 meters
Available energy	1 J
Transmission cost	600 mW
Receiving cost	200 mW
Clock frequency	2 MHz
Clock fluctuation	within $\pm$100 ppm
Transmission rate	1 Mbps
Signal propagation speed	$3 * 10^8$ meters/second
Processing delay (saturated)	0.05 seconds $\pm$ $N(0,1)$ msec access fluctuation
Processing delay (not saturated)	0.01 seconds $\pm$ $N(0,1)$ msec access fluctuation
Peer Evaluation Scan Message Length	98 bits
Peer Evaluation Reply Message Length	158 bits
Peer Evaluation Result Message Length	139 bits
Timing Information Message Length	152 bits
ACK Message Length	20 bits
Local ID Range	260
Mobility of Mobile Nodes	$8.3 * 10^{-4}$ meters/second
Mobility of Static Nodes	0 meters/second

In addition, the processing delays for both saturated and not saturated networks have access fluctuations that are normally distributed with mean and standard deviation of 0 and 1 msec. The access fluctuations are lumped values of both the medium access and software access fluctuations. As shown in Figure 28, the software access fluctuation of a Sparc machine running Solaris operating system is around 600 μsec while the mean access time is around 100 μsec. Experiments are run to test the medium access fluctuation of 802.11 MAC by running Windows 98 operating system in Compaq Presario and Sony laptops. The round trip fluctuations consisting forward and reverse medium accesses as well as software accesses are between 1 and 2 msec. As a result, the medium access fluctuation is in the order of few hundred μsec. Since

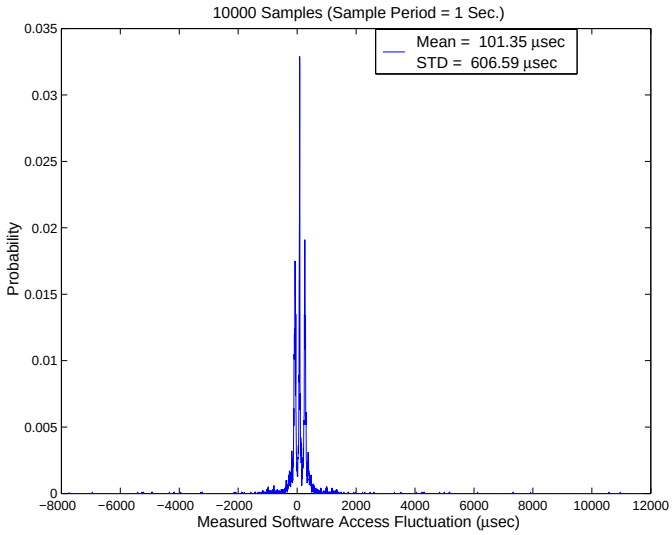

Figure 28: Software access fluctuation.

sensor nodes are designed for the low-end regime, the medium access and software access may fluctuate even more. For the simulations, the lumped access fluctuations are normally distributed as given in Table 8.

The performance of TDP is evaluated for both static and mobile sensor nodes by varying τ and setting δ and ρ values to 2 seconds and 0.3, respectively. First, the TDP is evaluated with and without the procedure PEP for both mobile and static nodes in Section 3.5.1. In addition, the TDP is compared to TPSN to show its novelties. As described in [26], TPSN performs better than RBS in single hop as well as multiple hops. As a result, the TDP is compared to TPSN in a network-wide scenario with parameters given in Table 8. Afterwards, both time convergence and energy dissipation of TDP are studied in depth for both static and mobile nodes in Section 3.5.2.

3.5.1 Performance Comparison

3.5.1.1 With/Without Peer Evaluation Procedure

As shown in Figure 29, the performance of TDP is evaluated for both with and without the procedure PEP for static and mobile nodes. The procedure PEP is designed to

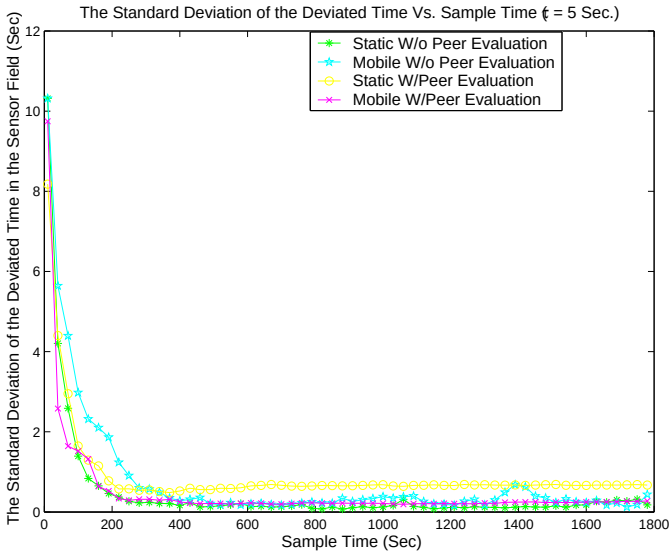

Figure 29: Comparison of time convergence.

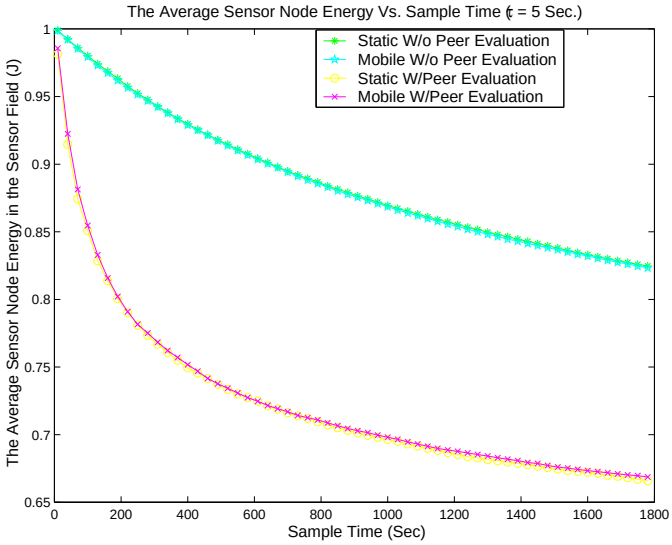

Figure 30: Comparison of energy consumption.

prevent the false tickers from participating in becoming master nodes. As the time throughout the sensor network converges, there is still a small time fluctuation within the network when the procedure PEP is not applied. This is illustrated in Figure 29 as the converged time wiggles after 400 seconds. On the other hand, the converged time is stable when the procedure PEP is used.

Although the procedure PEP provides a cleaner time throughout the network,

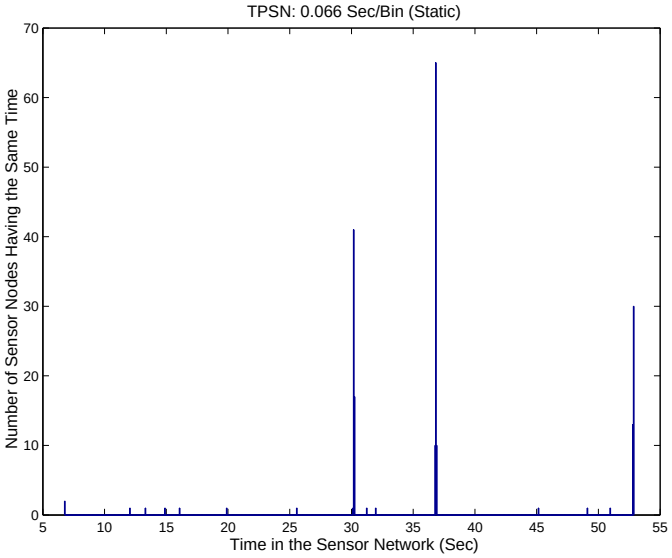

Figure 31: TPSN: histogram of time distributed in the network (static nodes).

the amount of energy consumed when the TDP is used with this procedure exceeds the TDP without it as shown in Figure 30. The TDP without the procedure PEP consumes 20% less energy, but the trade-off is allowing the time to fluctuate a little after convergence. If energy is more critical than time accuracy, the TDP without the procedure PEP may be a better choice. As a result, the performance of TDP without the procedure PEP is evaluated in detail in the following sections.

3.5.1.2 TDP vs. TPSN

The performance of TDP is compared with *Time-Sync protocol for Sensor Networks* (TPSN) [26] in a network-wide scenario to show how the diffusion process helps to synchronize the time in the network. For TPSN, there are three sinks trying to synchronize the network. After the nodes are synchronized, the histogram of the sensor nodes' time is calculated and shown in Figure 31. There are three large islands of time occurring approximately at 30 sec, 37 sec, and 54 sec. These islands of time are known to occur [26] when three sinks are used to synchronized the network. These islands of time may cause problems when the users want all sensor nodes to perform a task at a specific time. Although most of the sensor nodes are synchronized to either

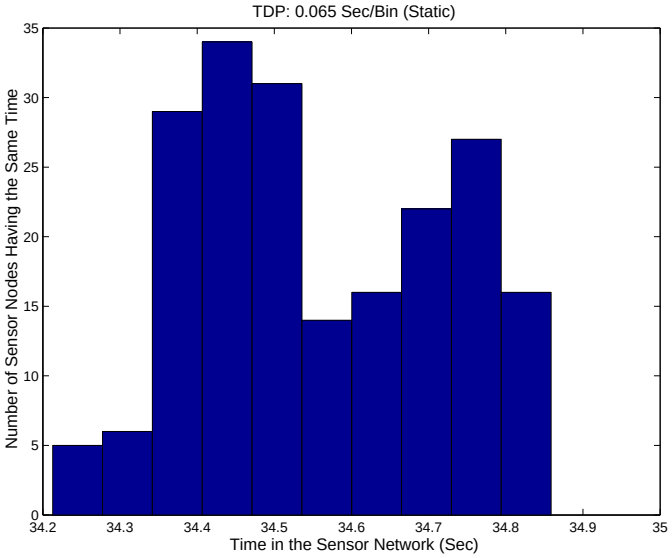

Figure 32: TDP: histogram of time distributed in the network (static nodes).

one of the three sinks, there are still some nodes that remain unsynchronized. From example, some of the sensor nodes have time values that are within the range of 5 sec and 27 sec. This anomaly may be due to (1) the broadcast radius not being large enough and (2) the timing offset of synchronization messages between two levels in the hierarchy.

Under the same simulation scenario, the TDP is applied. Since the TDP does not depend on specific sensor nodes to be master nodes, it enables the network time to reach an equilibrium value by diffusion process. As shown in Figure 32, the equilibrium time is around 34 sec. The time variation throughout the network is around 0.6 sec. This variation may be much tighter when the master nodes are synchronized to a time server.

When the sensor nodes are mobile, the TPSN exhibits more noise in the time throughout the network. Since TPSN synchronizes the nodes in the network hierarchically, the node movement breaks the hierarchy causing nodes to be unsynchronized. As shown in Figure 33, there are still three islands of time but more nodes are becoming unsynchronized due to the movements. As for TDP, the movement does not

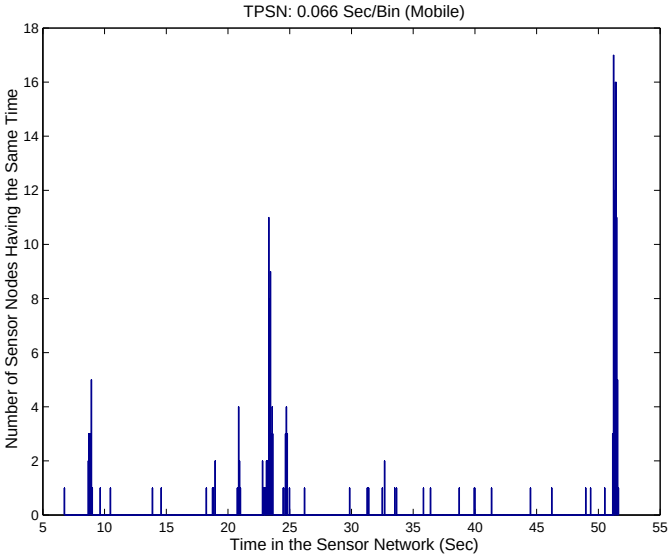

Figure 33: TPSN: histogram of time distributed in the network (mobile nodes).

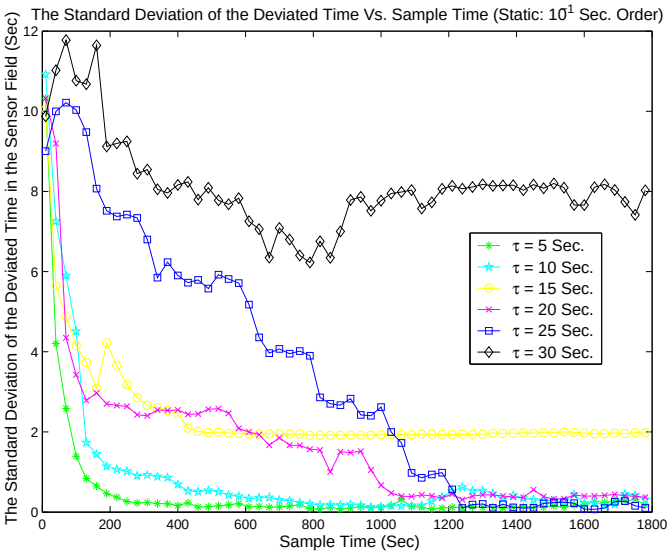

Figure 34: The standard deviation of time for different τ values (Static Nodes).

affect the diffusion process. The time throughout the network still reaches an equilibrium value. A more detailed evaluation of TDP is given in the following section. The performance of TDP over time is evaluated for both static and mobile nodes.

3.5.2 Performance Over Time

3.5.2.1 Static Sensor Nodes

The design objective of τ is to control the speed that the time in the sensor nodes reaches an equilibrium time. As shown in Figure 34, the convergence of time is illustrated for different values of τ. The standard deviation of the deviated time approaches 10^{-1} seconds order for $\tau = 5\ Sec$ and $\tau = 10\ Sec$. For $\tau = 15\ Sec$, the standard deviation flats out around 2 seconds. This is due to sensor nodes being topologically unaccessible with a broadcast radius of 10 meters. As for $\tau = 30\ sec$, the time in the sensor network fluctuates at a much faster rate than it can be synchronized.

The rate of the convergence depends on the τ value being used. The time in the sensor network converges the fastest when $\tau = 5\ sec$ as shown in Figure 34. This corresponds to the analytical performance evaluation. The convergence rate depends on the number of rounds of timing information message diffusion within a period of time. As a result, $\tau = 5\ Sec$ gives the highest number of rounds of timing information message diffusion.

The TDP is also aware of the energy consumed by the sensor nodes. It tries to evenly distribute the load to all the nodes as shown in Figure 35. The standard deviation of the sensor node energy in the sensor field is approaching a constant value. This means that the variation of node energy becomes constant.

To further show that TDP is performing as it should be, a 3-dimensional view of the mean square error (MSE) time distributed throughout the sensor field for $\tau = 5\ sec$ at simulation time 400 seconds is illustrated in Figure 36. A grid size of 8 meters by 8 meters is used to scan the whole sensor field. The grid is shifted at 1 meter increment horizontally and vertically until the whole sensor field is covered. After each grid movement, the MSE time of nodes within the grid are calculated. The MSE of the time deviated from the average is illustrated in Figure 36. It is in the $10^{-2} sec^2$ order. In the sensor network, the time difference between neighbor nodes is

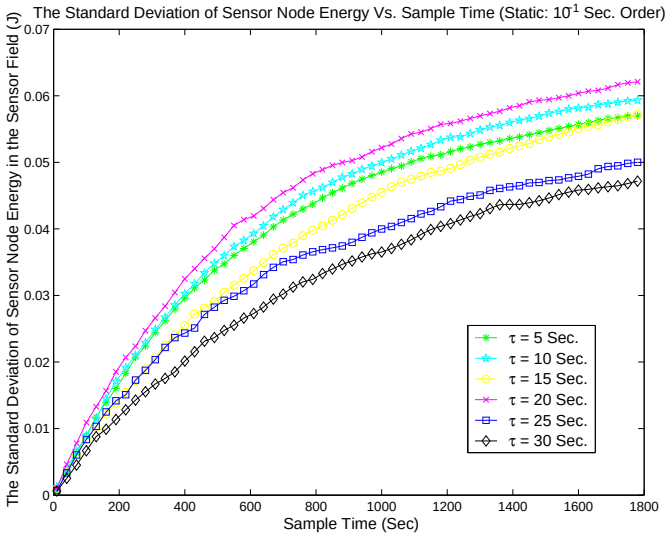

Figure 35: The standard deviation of energy for different τ values (Static Nodes).

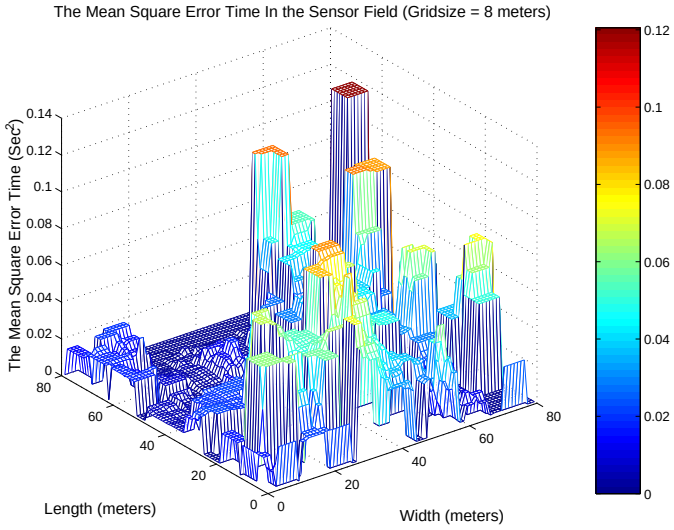

Figure 36: The MSE time of the network for $\tau = 5$ seconds (Static Nodes).

important for some applications, e.g., speed tracking. As a result, a smooth transition of time throughout the sensor field is important. The TDP does enable the time in the network to have a smooth transition as shown in Figure 36. At location (40,40) on the sensor field, the smooth transition is shown more prominently.

A detailed view of the MSE energy in the sensor field for $\tau = 5$ *sec* at simulation time 400 seconds is illustrated in Figure 37, where the energy variation is in 10^{-3} J^2 order. In addition, the energy is fairly distributed within the sensor field. The MSE

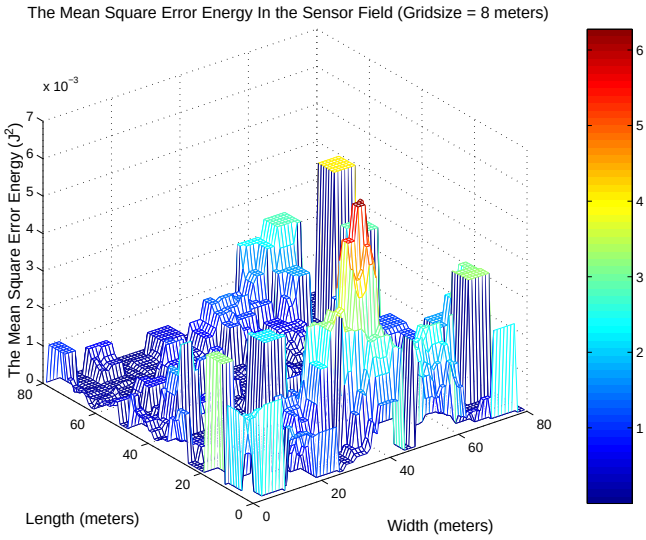

Figure 37: The MSE energy of the network for $\tau = 5$ seconds (Static Nodes).

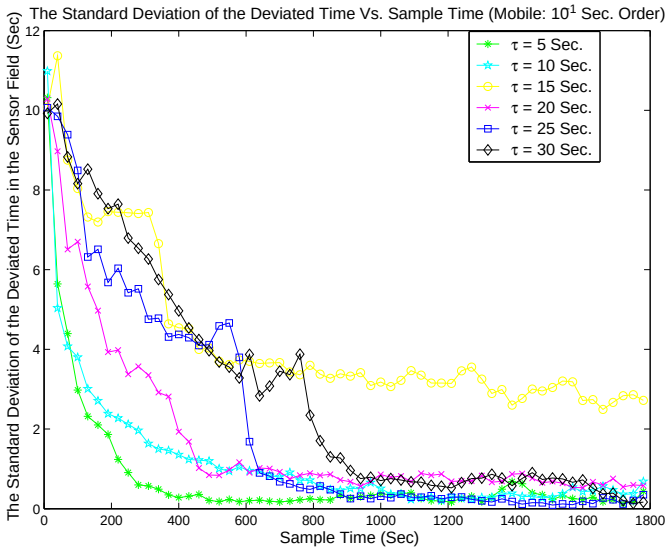

Figure 38: The standard deviation of time for different τ values (Mobile Node).

time and energy is highest at location (80,80). This is because the sensor nodes may not be easily accessible.

3.5.2.2 Mobile Sensor Nodes

When the sensor nodes are mobile, the time difference between the neighbor nodes can be higher than when the nodes are static. As a result, the frequent change of positions and sharper time difference among neighbors cause the TDP to converge

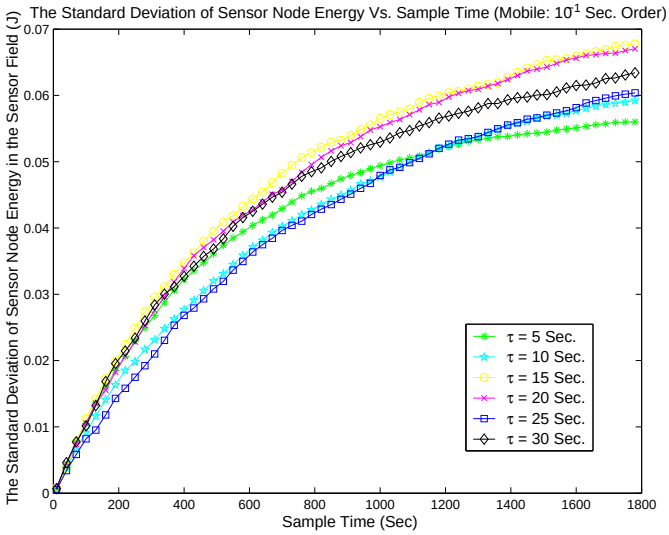

Figure 39: The standard deviation of energy for different τ values (Mobile Nodes).

slower as shown in Figure 38. As the τ value increases, it takes a longer time to converge. In addition, the converged time is at a higher value for large τ values. It is because the timing information message diffusion rate can not keep up with the mobility of the nodes. This suggests that the τ value has to be small for high mobility or else the time in the network will have a hard time to converge. For instance, the time converges at around 400 sec for $\tau = 5$ *sec* while it is at 1700 sec for $\tau = 30$ *sec* as shown in Figure 38. When $\tau = 15$ *sec*, the time converges to around 3 sec. This is due to nodes not topologically accessible with a broadcast radius of 10 meters.

The load of participating in the TDP is also distributed to all the nodes as shown in Figure 39. The standard deviation of a sensor node energy in the sensor field is slowly approaching a constant value over time. Regardless of the time in the network converges or not, the energy consumption of the nodes are fairly distributed.

Chapter 4

Distributed Perceptive Localization Framework for Sensor Networks

Lastly, the ability to provide an effective way of finding the location of nodes is vital to many applications, such as target tracking and environmental monitoring. The distributed perceptive localization framework addresses this by allowing users to find the nodes using purely relative positions, i.e., without fixed base stations. The motivation for the work and the details of the localization technique are discussed in the following sections.

4.1 Motivation and Related Work

The application of many sensor nodes to track the location of an event or target is appealing and has caught the attention of many researchers. The vision is to use many low-end sensor nodes to provide location information that a traditional single sensor may not be able to provide. For example, the path of a moving target may be easier to obtain with many low-end sensor nodes than a single high-end sensor. Since the sensor nodes may be much cheaper than a traditional single sensor, the computational power, sampling rate, and circuit precision are lower causing non-Gaussian noise in range and angle estimations [20]. In addition, sensor nodes may be deployed in areas where beacons [61][49] or GPS are not applicable and the networks may be frequently jammed by environmental or manually induced noise.

The *Ad-Hoc Localization System* (AHLoS) [61] requires few nodes which have known location either through GPS or manual configuration. AHLoS allows sensor nodes to discover their location through a two-phase process: *ranging* and *estimation*.

During the ranging phase, each node estimates the range of its neighbors. The estimation phase then allows neighbors without location information to use this range estimate and the known location of the beacons in order to determine their locations.

Also, methods presented in [9][10] assume that beacon signals at certain known locations are available. This assumption may be fine for some applications, but sensor nodes may be deployed in some regions where known locations may not be possible.

As a result, self-localization using sources at unknown locations is being investigated [49]. Although [49] relaxes the assumption that beacons require fixed locations, it still needs to have a number of signal sources. These signal sources are deployed in the same region as the sensor nodes and are used as references by the neighboring nodes to estimate the unknown locations and orientations from the signal sources.

In addition to these localization methods [61][9][10][49], [55] provides the theoretical Cramér-Rao Bound (CRB) of the localization errors. The CRB gives the lower error bound, and it is an important baseline on how well the localization methods are performing.

As the state-of-the-art [61][9][10][49] uses signal sources to self-calibrate, there is still a need to have localization methods that do not require signal sources. There are few reasons for this: (1) signal sources may die or may be out-of-range, (2) some environments such as underground caves can not use GPS, and (3) low-end sensor nodes may have nonlinear characteristics and significant amount of noise when performing range or angle estimations.

The goal of this chapter is to introduce a *perceptive localization framework* (PLF) that enables a node to detect and track the location of the neighboring node–allowing distributed localization. The PLF consists of a collaborative estimation technique and a particle filter. Its purpose is to address the noise and non-GPS issues in localization that are applicable to some sensor network applications.

In the collaborative estimation technique, the sensor nodes collaborate to estimate

the relative positions of their neighboring nodes, because the sensing units used by the sensor nodes may have significant amount of noise. This noise causes the errors in range and angle estimations to increase with respect to distance. Once the relative positions of the nodes are obtained, the position of any node with respect to the sink may be calculated by summing all the relative positions along the route between the node and sink. To increase the accuracy of the location estimation, the sensor nodes along the route may increase the number of samples (particles) used by the particle filter. This process of local interaction does not require any beacons in place. In addition, a central processing unit is not required in order to determine the locations of the sensor nodes.

Furthermore, theoretical lower error bounds of the collaborative estimation techniques are provided, and simulations are conducted to show the validity of the collaborative estimation techniques and position tracking via a particle filter. The contributions of PLF are as follows:

1. *It allows users to locate sensor nodes using hop-by-hop relative positions.*

2. *It lowers the energy cost in determining the relative positions since it uses a particle filter to track the neighboring nodes.*

3. *It enables the increase of localization accuracy when the number of samples or the number of nodes deployed increases.*

4. *It is robust to sensor node failures.*

This chapter is organized as follows: In Section 4.2, a vision of a sensor node design is provided. Afterwards, the PLF is described in Section 4.3, and the end-to-end localization error is discussed in Section 4.4. A method using the PLF to locate a node in the sensor field is described in Section 4.5, and the performance evaluation is discussed in Section 3.5.

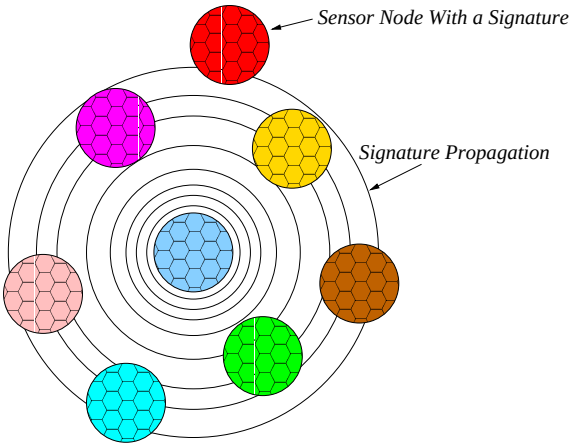

Figure 40: The propagation of signature.

4.2 Envision of Sensor Nodes

Sensor nodes may be deployed in areas where people and vehicles are not able to traverse, e.g., chemically hazardous environment, deep sea, and underground water channels. In addition, the Global Positioning System (GPS) may not be able to function properly in these or other areas. As a result, the sensor nodes may loose their sense of location. One possible design of a sensor node may consist some of the following features: (1) multiple antennas, (2) large number of sensing units, (3) sphere shape, and (4) multiple signatures that can be generated thermally, acoustically, chemically, electrically, electro-magnetically, or magnetically. A sensor node may use multiple antennas to increase the reception capability and efficiency. In addition, a large number of sensing units makes it easy to determine the orientation of the neighboring node without using complex angle of arrival techniques. The locations of these sensing units on the sensor node are known aprior by the sensor node. A sensor node just have to determine the sensing unit that gives the best signal when signatures are propagated by the neighboring nodes as shown in Figure 40.

A sensor node may have one or multiple signatures, because different types of signatures are needed for different applications. For example, light sensors can be used for airborne explorations (i.e., studying the chemical compositions in air) and

thermal sensors for cave explorations. The set of signatures that the sensor nodes can choose from has to be large enough so that the neighbors have a very low probability of having the same signature.

These proposed features allow the sensor nodes to better estimate the relative positions of their neighboring nodes, so a distributed perceptive localization framework is feasible. This framework specifies how a node estimates the relative positions of their neighboring nodes, and it is described in detail in Section 4.3. Once the relative positions of the nodes are obtained, the position of any node with respect to the sink may be calculated by summing all the relative positions along the route between the node and sink. Thus, GPSs or beacons are not needed for localization.

4.3 Perceptive Localization Framework (PLF)

A node in the sensor field can be located exactly by another node if all the nodes could know the exact relative positions of their neighbors. Current solutions based on as *received signal strength* (RSS) [67][45] and *time-of-arrival* (TOA) [25][66] can be used to determine the relative positions between neighboring nodes. Since sensor nodes are lower in cost, power, and computation abilities than traditional ad hoc devices [2], it is a challenge to achieve a good estimate of the relative positions using low-end devices that have high noise, low signal sensitivity, and low computational capability. Thus, the objective of our work is to address this challenge by proposing the perceptive localization framework (PLF).

The PLF consists of two components as shown in Figure 41: *collaborative estimation* (CE) and *particle filtering* (PF). The CE component determines the initial position of the neighboring node while the PF component is used to track the position of the node if the node is mobile. As a result, the neighboring node may be static or mobile. If it is static, only the CE component is used since the position tracking is not needed. Otherwise, the PF component is used to determine the relative position

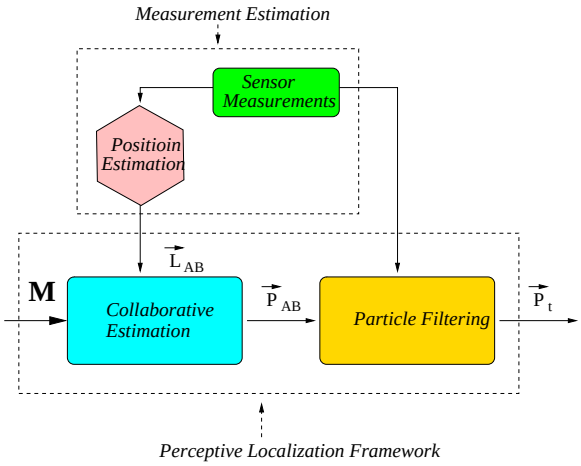

Figure 41: The perceptive localization framework.

of the neighboring node by sensing and tracking instead of transmitting messages to conserve energy.

For example, let us assume node A is perceiving the position of neighboring node B. As shown in Figure 41, the CE component provides the estimated position $\vec{P}_{AB}$ of B to the PF. The inputs to the CE component are $\vec{L}_{AB}$ and $\mathbf{M}$; $\vec{L}_{AB}$ is the position estimation of B by A using the sensor measurement data while $\mathbf{M}$ is a set of nodes that are in the sensing range of B. The position estimation $\vec{L}_{AB}$ may be determined by existing RSS or TOA techniques. For instance, node A uses the RSS technique to estimate the distance of B from A by measuring the signal strength transmitted by B. On the other hand, the time difference of two signal pulses can be used to determine the relative distance if TOA technique is used. This type of estimation (RSS or TOA) is termed as *measurement estimation* (ME) as shown in Figure 41. Once the PF component receives the estimated position $\vec{P}_{AB}$, it uses the sensor measurement data to track the position of B. The output is the estimated position $\vec{P}_t$ of B at time t.

The accuracy of the PLF is based on (i) *the number of sensing units a node has*, (ii) *the quality of the sensing units*, (iii) *the density of sensor nodes*, and (iv) *the number of samples used for PF*. These factors are explained below.

- *The Number of Sensing Units A Node Has*: When there are many sensing units, the error in estimating the relative position is lower. As described in Section 4.2, a sensor node can determine the direction of the neighboring node by identifying the sensing unit that gives the best signal when measuring the signature. With many sensing units, the spacing between them is closer. Thus, the error in estimating the direction is lower.

- *The Quality of the Sensing Units*: The quality (noise level, signal sensitivity, and temperature tolerance) of the sensing units heavily affects the estimation accuracy of the CE and PF components. A better quality sensing unit has less measurement noise.

- *The Density of the Sensor Nodes*: The position estimation accuracy of the CE component increases with the node density. The CE algorithm is designed so that a node collaborates with its neighbor nodes to obtain a better estimate of the relative position.

- *The Number of Samples Used for PF*: As the number of samples used for the particle filter increases, the position tracking accuracy also increases. If the number of samples is large, the computation cost may be high. As a result, care must be taken when selecting the size of the samples; it is a trade-off between computation cost and tracking accuracy.

The CE and PF components of the PLF are discussed in Sections 4.3.1 and 4.3.2, respectively. Afterwards, the PLF is used to locate a node in the sensor field, which is described in Section 4.5.

4.3.1 Collaborative Estimation

As stated in Section 4.1, one of the purposes of PLF is to address the noise exhibited by the low-end devices when performing range estimations. There are two techniques

to estimate the range: *received signal strength* (RSS) [67][45] and *time of arrival* (TOA) [25][66]. The theoretical lower error bounds for both techniques are presented in Section 4.3.1.1. In addition, new theoretical error bounds are calculated when RSS or TOA is used in collaborative estimation. Afterwards, an implementation of the collaborative estimation is presented in Section 4.3.1.2. It tries to reach the new theoretical lower error bounds.

4.3.1.1 Collaborative Estimation: Theoretical Lower Error Bounds

As mentioned in Section 4.3, the error in estimating the direction of the neighboring nodes lowers when a node has many sensing units. The directional error is assumed to be small. Hence, the focus of this paper is to estimate the relative position of the neighbor nodes by reducing the error in the estimated distance. The TOA or RSS technique may be used to estimate the distance between neighboring sensor nodes, e.g., A and B. The error for the TOA technique is zero mean Gaussian with probability density function [55] as calculated by

$$f_{T_{A,B}|d_{A,B}}(T_{A,B}|d_{A,B}) = \frac{1}{\sqrt{2\pi\sigma_T^2}}\, e^{\left(-\frac{(T_{A,B}-d_{A,B}/c)^2}{2\sigma_T^2}\right)} \tag{21}$$

where $T_{A,B}$ is the measured time delay in seconds from node B to A; σ_T is the standard deviation of the measured time delay; and $d_{A,B}$ is the distance between nodes A and B.

As for the RSS technique, the probability density function of the zero mean Gaussian is determined by

$$f_{P_{A,B}|d_{A,B}}(P_{A,B}|d_{A,B}) = \frac{1}{\sqrt{2\pi\sigma_{dB}^2}}\, e^{\left(-\frac{(P_{A,B}-\bar{P}_{A,B})^2}{2\sigma_{dB}^2}\right)}$$

$$\bar{P}_{A,B} = P_0 - \frac{10n \ln\left(\frac{d_{A,B}}{d_0}\right)}{\ln 10}, \tag{22}$$

where $P_{A,B}$ is the power measured in dBm at sensor node A transmitted by sensor node B; σ_{dB} is the standard deviation of the measured power; P_0 is the received power at a reference distance d_0; and n is the path loss exponent.

The Cramér-Rao Bound (CRB) [17] of an unbiased estimator of $d_{A,B}$ gives the lower bound of the estimation error. The CRB uses the Fisher Information Matrix (FIM) to calculate the lower error variance of the estimator, which can either be a TOA or RSS technique. The CRBs of the error variances $\sigma^2_{d_{A,B}}$ for both TOA and RSS techniques are derived and calculated in Appendix B with the probability density functions given by (21) and (22). They are presented as follows:

$$TOA : \sigma^2_{d_{A,B}} \geq \sigma^2_T c^2, \tag{23}$$

$$RSS : \sigma^2_{d_{A,B}} \geq \left(\frac{d_{A,B}\ \sigma_{dB}\ \ln 10}{10n}\right)^2. \tag{24}$$

These bounds (23) and (24) provide the minimum estimation error achievable between nodes A and B. The bound for the RSS technique (24) varies with the distance $d_{A,B}$ while it is independent of the distance for the TOA technique (23). Hence, the TOA technique may have a better error bound than the RSS technique. On the other hand, if the distance between sensor nodes is reduced by deploying more sensor nodes, the RSS technique may also be a good alternative. It is cheaper and easier to build than the TOA technique since the TOA requires higher sampling rate to receive the signal. Thus, different types of sensor network applications can choose between RSS or TOA technique depending on the cost and complexity of the electronics.

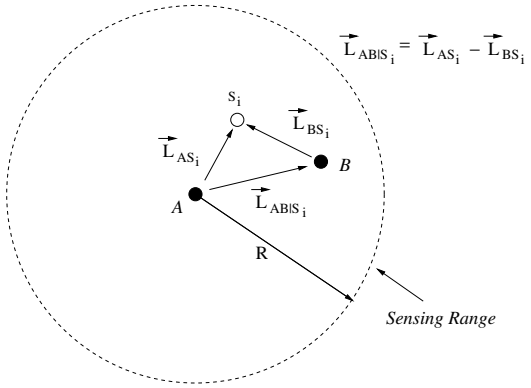

Figure 42: The calculation of $\vec{L}_{AB|S_i}$.

Since the CRBs of the TOA and RSS techniques are determined by (23) and (24), a collaborative estimation method based on either TOA or RSS technique is proposed to lower the CRB. The collaborative estimation method lowers the CRB by allowing a sensor node S_i that is within the sensing range of sensor nodes A and B to help estimate the distance between nodes A and B. As shown in Figure 42, sensor node A estimates the distance $d_{A,S_i} = ||\vec{L}_{AS_i}||$ between nodes A and S_i while sensor node B estimates the distance $d_{B,S_i} = ||\vec{L}_{BS_i}||$. The relative position $\vec{L}_{AB|S_i}$ of sensor node B from A given node S_i is estimated by summing the vectors $\vec{L}_{AS_i}$ and $\vec{L}_{S_iB}$ as shown in Figure 42.

The reason to use an intermediate node S_i to help estimate the distance $d_{A,B} = ||\vec{L}_{AB|S_i}||$ between sensor nodes A and B is because the CRB of the RSS technique (24) increases with distance $d_{A,B}$. In addition, the variance σ^2_{dB} may change with respect to distance in a real environment (actual deployment), which is shown in Section 4.6.1 with performance results obtained from a physical testbed. Although the CRB of the TOA technique (23) does not change with respect to distance, the variance σ^2_T may still fluctuate in a real environment causing the CRB to change with distance.

Let's assume that the variances σ^2_T and σ^2_{dB} change with respect to distance. The CRB $\sigma^2_{CRB,T}$ of the TOA technique with collaborative estimation is given by

$$TOA : \sigma^2_{CRB,T} = c^2\Big(\sigma^2_{T,d_{A,S_i}} + \sigma^2_{T,d_{B,S_i}}\Big), \tag{25}$$

where $\sigma^2_{T,d_{A,S_i}}$ and $\sigma^2_{T,d_{B,S_i}}$ are the variances at distance d_{A,S_i} and d_{B,S_i}, respectively. The CRB $\sigma^2_{CRB,T}$ (25) is the summation of the CRBs from sensor node A to S_i and node B to S_i.

Furthermore, the CRB $\sigma^2_{CRB,dB}$ of the RSS technique with collaborative estimation is defined as

$$RSS : \sigma^2_{CRB,dB} = \beta\left(\frac{\ln 10}{10n}\right)^2, \tag{26}$$

where β is calculated by

$$\beta = (d_{A,S_i}\ \sigma_{dB,d_{A,S_i}})^2 + (d_{B,S_i}\ \sigma_{dB,d_{B,S_i}})^2. \tag{27}$$

The distances d_{A,S_i} and d_{B,S_i} are from node A to S_i and B to S_i, respectively. In addition, $\sigma^2_{dB,d_{A,S_i}}$ and $\sigma^2_{dB,d_{B,S_i}}$ are the variances at distances d_{A,S_i} and d_{B,S_i}, respectively. Thus, (26) is the summation of the CRBs from sensor node A to S_i and node B to S_i.

To give an idea of how the CRBs $\sigma^2_{CRB,T}$ and $\sigma^2_{CRB,dB}$ of the TOA and RSS techniques behave with respect to the position of node S_i, (25) and (26) are plotted in Figures 43 and 44 with $c = 3 * 10^8 m/s$, $\sigma^2_{T,d_{A,S_i}} = (d_{A,S_i}/R) * \sigma^2_T$, $\sigma^2_{T,d_{B,S_i}} = (d_{B,S_i}/R) * \sigma^2_T$, $\sigma_T = 6.1\ ns^2$, $n = 2.30$, $\sigma_{dB} = 3.92\ dB$, $\sigma^2_{dB,d_{A,S_i}} = (d_{A,S_i}/R) * \sigma^2_{dB}$, $\sigma^2_{dB,d_{B,S_i}} = (d_{B,S_i}/R) * \sigma^2_{dB}$, node A at (0,5), and node B at (10,5).

As shown in Figures 43 and 44, the CRBs $\sigma^2_{CRB,T}$ and $\sigma^2_{CRB,dB}$ of both TOA and RSS techniques concave toward the center of nodes A and B located at (0,5) and (10,5). For instance, the lowest value of the CRB $\sigma^2_{CRB,T}$ lies on the line between sensor nodes A and B while the lowest value of CRB $\sigma^2_{CRB,dB}$ is located at (5,5).

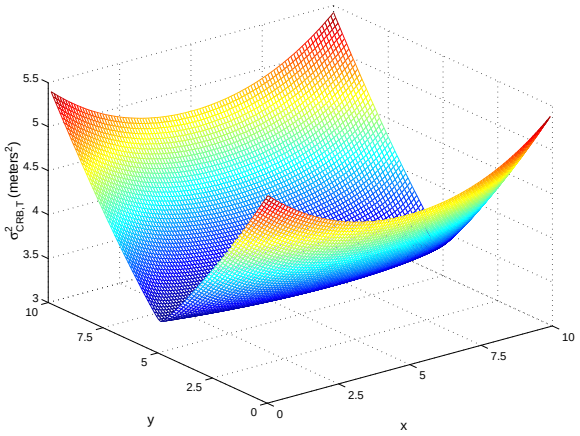

Figure 43: The $\sigma^2_{CRB,T}$ with respect to the location of S_i.

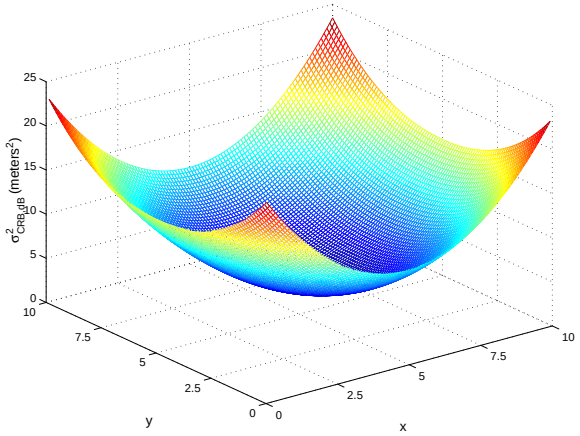

Figure 44: The $\sigma^2_{CRB,dB}$ with respect to the location of S_i.

The performances in Figures 43 and 44 are expected since the CRBs $\sigma^2_{CRB,T}$ and $\sigma^2_{CRB,dB}$ are lower when the distances d_{A,S_i} and d_{B,S_i} are smaller. Comparing the performances of both TOA and RSS techniques, the TOA technique has a lower CRB value than the RSS technique as shown in Figures 43 and 44, but it may require more costly hardwares to sample the received signal. Both methods are feasible, and the choice between the two depends on the application of the sensor networks.

4.3.1.2 Collaborative Estimation: Implementation

The CRBs $\sigma^2_{CRB,T}$ (25) and $\sigma^2_{CRB,dB}$ (26) of the TOA and RSS techniques provide an insight into implementing the collaborative estimation (CE) algorithm. According to

Collaborative Estimation

Given: **M** *and* $\vec{L}_{AB}$

Steps:

(1) Determine set **T** *, which is a set of nodes in the sensing range of node A*

(2) $\mathbf{S} = \mathbf{M} \cap \mathbf{T}$ *where* $\mathbf{S} = \{ S_1, S_2, S_3, ..., S_k \}$

(3) if ($\mathbf{S} = \phi$ *)*

$$\vec{P}_{AB} = \vec{L}_{AB}$$

else

(a) $\vec{L}_{AB|S_i} = \vec{L}_{AS_i} - \vec{L}_{BS_i}$ *for i=1 to k*

(b) $\alpha_i = R / \| \vec{L}_{AS_i} \|$ $\beta_i = R / \| \vec{L}_{BS_i} \|$ *for i=1 to k*

(c) $\alpha_T = \sum_{i=1}^{k} \alpha_i$ $\beta_T = \sum_{i=1}^{k} \beta_i$

(d) $\rho_i = \frac{(\alpha_i / \alpha_T + \beta_i / \beta_T)}{2}$ *for i=1 to k*

(e) $\vec{P}_{AB} = \sum_{i=1}^{k} \rho_i \vec{L}_{AB|S_i}$

end

(4) Return $\vec{P}_{AB}$

Figure 45: The collaborative estimation algorithm.

Figures 43 and 44, the minimum values of $\sigma^2_{CRB,T}$ and $\sigma^2_{CRB,dB}$ lie between the sensor nodes A and B. Thus, the CE algorithm should give more weight to the estimated relative position $\vec{L}_{AB|S_i}$ calculated with sensor node S_i that lies near the middle of the sensor nodes A and B.

The CE algorithm is illustrated in Figure 45. It tries to reach the minimum values of the CRBs $\sigma^2_{CRB,T}$ (25) and $\sigma^2_{CRB,dB}$ (26). First, it determines the set **T** that contains the nodes in the sensing range of node A. Since the set **M** contains nodes within the sensing range of B, the intersection of sets **M** and **T** is given by set **S** in step 2. The intersection of **M** and **T** is shown graphically in Figure 46; R is the sensing range of nodes A and B. The shaded region contains the nodes in set **S**. The nodes in set **S** are used to help determine the relative position of B from A. As shown theoretically in Section 4.3.1.1, the CRBs for both TOA and RSS techniques are lowered when collaborative estimation is used. Thus, the nodes in set **S** collaboratively improves

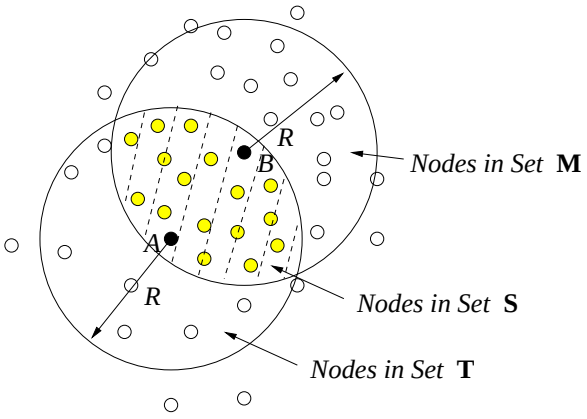

Figure 46: The intersection of set **M** and **T**.

the estimation accuracy.

If set **S** is empty (null), then $\vec{P}_{AB}$ is equal to $\vec{L}_{AB}$. This is to ensure that there is an estimated position of B for the PF component as specified in Figure 41, and the performance of CE should be equal or better than the measurement estimation, ME, which just estimates the relative position of node B without the help from an intermediate node S_i. If the set **S** does contain nodes, then the estimated position $\vec{L}_{AB|S_i}$ of B given node S_i is calculated by step (3.a) and illustrated in Figure 42. The radius R as shown in Figure 42 is the sensing range of the nodes. The perceived position of B from A may be determined by using the neighbor node S_i as an intermediate node to help refine the relative position estimation. Since node A may be far from B, node S_i, which is in-between nodes A and B, may offer a better estimate of $\vec{L}_{AB}$. Nodes that reside in the middle of the nodes A and B provide the best estimate of the relative position $\vec{L}_{AB}$, because the CRBs $\sigma^2_{CRB,T}$ (25) and $\sigma^2_{CRB,dB}$ (26) are the lowest when the node S_i is in the middle of the nodes A and B as explained in Section 4.3.1.1.

As a result, the weights for the $\vec{L}_{AB|S_i}$ estimations calculated with the nodes in set **S** are determined by steps (3.b), (3.c), and (3.d). They are inverse proportional to the distances $||\vec{L}_{AS_i}||$ and $||\vec{L}_{BS_i}||$. The weights α are measuring the contributions of the estimations as an inverse of the distance $||\vec{L}_{AS_i}||$ while the weights β are measuring as

an inverse of the distance $||\vec{L}_{BS_i}||$ (steps (3.b) and (3.c)). The nodes in set **S** that are closer to A and B are assumed to provide better $\vec{L}_{AB|S_i}$ estimations since estimation error may increase rapidly as a function of distance. After the weights ρ are calculated in step (3.d), the weighted estimation $\vec{P}_{AB}$ of the relative position is calculated in step (3.e) and sent to the PF component as shown in Figure 41. The weights ρ are giving more emphasis on $\vec{L}_{AB|S_i}$ estimations contributed by sensor nodes S_i that are near the middle of sensor nodes A and B.

The message overhead and computational complexity of the CE component as shown in Figure 45 are $O(|\mathbf{T}| + |\mathbf{M}| - |\mathbf{S}|)$ and $O(|\mathbf{S}|)$, respectively. They grow with respect to the number of sensor nodes within the sensing radius R. Although the message overhead and computational complexity are higher for denser sensor networks, the estimation accuracy is shown to improve via simulation in Section 4.6.2.

4.3.2 Particle Filtering

As described in Section 4.3.1, the CE component is used to estimate the relative position of the neighboring node when the nodes are static. When nodes are moving, the CE component is used to provide the initial relative position estimation of the neighboring node while the particle filtering (PF) component is used to track the position of the neighboring node.

Particle filters [20] are used to estimate the state **P** of a dynamic system based on the observation **U**. The state space of tracking a neighboring node is the Cartesian coordinates (x,y,z). The particle filter approximates the posterior probability distribution of **P** at time t by N weighted samples (particles) of the proposal probability distribution. The proposal probability distribution is an estimate of the posterior probability distribution. The proposal distribution that we used is the transition prior $q(\mathbf{P_t}|\mathbf{P_{t-1}})$ [4][6][28][38][42], where $\mathbf{P_t}$ is the state **P** at time t. Other proposal

distributions may be used, but they may require higher processing power and time. For sensor networks, a simple particle filter may be a good choice.

The state of the system $\mathbf{P_t}$ and output observations $\mathbf{U_t}$ at time t used by the particle filter are calculated as

$$\mathbf{P_t} = f(\mathbf{P_{t-1}}) \tag{28}$$

$$\mathbf{U_t} = g(\mathbf{P_t}) + n(\mathbf{P_t}) \tag{29}$$

where f(·), g(·), and n(·) are the movement, observation, and measurement noise models. For example, the output observations $\mathbf{U_t}$ may be the received signal strength or the time difference of arrival of the received signals, etc. Furthermore, there is no process noise since the objective is to track the natural movement of a neighboring node. In addition, there is no input observation (control signal) into (29). The details of a particle filter design and implementation are discussed in [46], and the basic procedures of a particle filter applied in tracking the location of the neighboring node are as follows:

1. *Initialization*: For i=1:N , set $\mathbf{P_0^i}$ to $\vec{P}_{AB}$ with Gaussian noise.

2. For t = 1, 2, 3, ...

   (a) *Importance Sampling*: For i=1:N, estimate $\mathbf{P_t^i}$ by $\widehat{\mathbf{P}}_\mathbf{t}^\mathbf{i}$, calculate the importance weights w_t^i, and normalize w_t^i

   (b) *Resampling*: For i=1:N, adjust $\widehat{\mathbf{P}}_\mathbf{t}^\mathbf{i}$ and w_t^i

   (c) *Output*: $\widehat{\mathbf{P}}_\mathbf{t}^\mathbf{i}$ is an approximation of the posterior distribution; $\mathbf{P_t}$ is the mean of $\widehat{\mathbf{P}}_\mathbf{t}^\mathbf{i}$.

The computational complexity of the particle filter is $O(N)$, which means that the complexity grows linearly with respect to the number of particles used. As previously

explained, the CE and PF components are used by the PLF to determine the relative position of the neighboring nodes. In Section 4.5, the PLF is used to find nodes in the sensor field, but first the end-to-end localization error is discussed in the following section.

4.4 End-to-End Localization Error

Since the PLF provides a framework to find the relative location of the neighboring nodes, it is important to know the end-to-end localization error between sink C and source D. The CRB of the end-to-end localization error is the sum of the CRB contributed by each hop, which is determined by (25) or (26). As a result, the variances σ^2_{CD} of the localization errors at source D with respect to the sink C are constrained as follows for the TOA and RSS techniques:

$$TOA : \sigma^2_{CD} \geq \sum_{j=1}^{o} \sigma^2_{CRB,T}(j), \tag{30}$$

$$RSS : \sigma^2_{CD} \geq \sum_{j=1}^{o} \sigma^2_{CRB,dB}(j), \tag{31}$$

where $\sigma^2_{CRB,T}(j)$ and $\sigma^2_{CRB,dB}(j)$ are the CRBs at hop j between a pair of nodes that are on the route.

The number of hops o in (30) and (31) depends on the size and density of the network in addition to the broadcast radius of the nodes. Thus, the distance between neighboring nodes may decrease when the density of the network increases. Also, the number of hops o from the sink C to the source D may increase.

4.5 Locating a Node in the Sensor Field

If a node C, e.g., a sink, wants to send a message to node D, e.g., a source, located at location (x_{CD},y_{CD},z_{CD}), it sets the targeted location (x_{CD},y_{CD},z_{CD}) in the message.

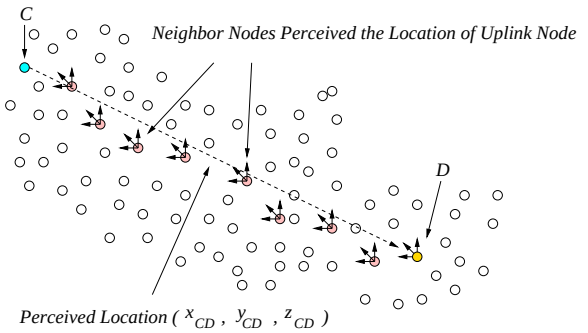

Figure 47: The perceived location of node D by node C.

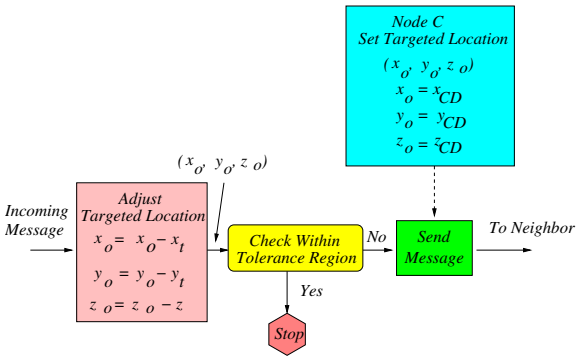

Figure 48: The tracing along the route to find node D; $\mathbf{P_t} = (x_t, y_t, z_t)$.

The location (x_{CD},y_{CD},z_{CD}) is the perceived location of the node D by node C as shown in Figure 47. The procedure of finding the targeted location is illustrated in Figure 48. The message is broadcast by the sink to its neighbors. The neighboring nodes adjust the targeted location (x_o, y_o, zo) in the message according to their perceived distance and direction, i.e., $\mathbf{P_t} = (x_t, y_t, z_t)$, of the node that originated the message. After the targeted location (x_o, y_o, zo) is adjusted as shown in Figure 48, the nodes check if the targeted location is within the tolerance region by

$$x_o^2 + y_o^2 + z_o^2 < \tau^2 \tag{32}$$

where (x_o,y_o,z_o) is the targeted location; o is the hop number; and τ is the radius of the sphere that represents the tolerance region. The tolerance region approximates the error accumulated from the sink C to source D, where each hop is Gaussian distributed. If the targeted location is within the tolerance region, the rebroadcasting

of the message stops. If not, the procedure continues hop-by-hop until the source is reached. The use of tolerance region is to enable nodes that are within the vicinity of the targeted location to respond to the sink. Sometimes when the nodes are randomly deployed, they may not fall exactly on the location specified by the sink.

Since the CRBs of the end-to-end localization errors for TOA and RSS techniques are given by (30) and (31), the minimum values for radius τ are determined as

$$TOA : \tau = \sqrt{\frac{\sum_{j=1}^{o} \sigma_{CRB,T}^2(j)}{3}}, \tag{33}$$

$$RSS : \tau = \sqrt{\frac{\sum_{j=1}^{o} \sigma_{CRB,dB}^2(j)}{3}}, \tag{34}$$

assuming the error is equally propagated into the x, y, and z directions. Thus, the radius τ is obtained by dividing (30) and (31) by 3 and taking the square root.

4.6 Performance Evaluation

The performance evaluation of the PLF between sensor nodes A and B (Figure 46) is separated into four parts. These four parts are *physical testbed*, *performance of CE*, *position tracking*, and *localization error*, which are described in Sections 4.6.1, 4.6.2, 4.6.3, and 4.6.4, respectively. The physical testbed subsection is to study the behavior of a cheap sensor (a Lapel EM-1 microphone) attached to node A and an actuator (a radio shack 15mA max @ 3 VDC mini buzzer with the test equipments as shown in Figure 49) attached to node B. The Lapel EM-1 microphone and 15mA max @ 3 VDC mini buzzer cost only few dollars; if they are integrated into a sensor node, the price may cost less than a dollar. The purpose is to study the signal energy attenuation and variation with respect to distance of low-end devices. This setup is to implement a RSS technique. We choose to study the RSS technique, because it is easier and cheaper to build a RSS device than a TOA device, which requires

Figure 49: The testbed.

a higher sampling rate to capture signals at speed-of-light resulting in an expansive component.

After the data are collected from the testbed, the observation model g(·) and measurement noise model n(·) are derived from the testing results. They are used to test the performance of the CE and PF components in Sections 4.6.2 and 4.6.3. In addition, sine and random way point movement models f(·) are used to test the ability of the PF component in Section 4.6.3. After the performance results are obtained for CE and PF components, the localization error in locating a node in the sensor field is analyzed and discussed in Section 4.6.4.

4.6.1 Physical Testbed

A 50/50 8Hz 3V square wave from an Instek GSG 8210 function generator is sent to a radio shack 15mA max @ 3 VDC mini buzzer for 60 seconds. The buzzer at node B is at multiples of 1/6 meters from the Lapel EM-1 microphone at node A, where the maximum distance between nodes B and A is 1.5 meters. The burst signal is captured by Matlab running on Presario 17XL260 sampling at 1KHz. The average received signal energy of the burst is illustrated in Figure 50. The received signal energy g(·), which is the observation model, as a function of distance can be approximated by

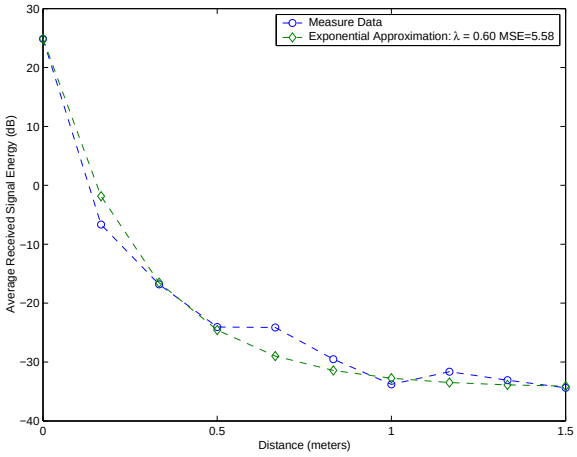

Figure 50: The average received signal energy.

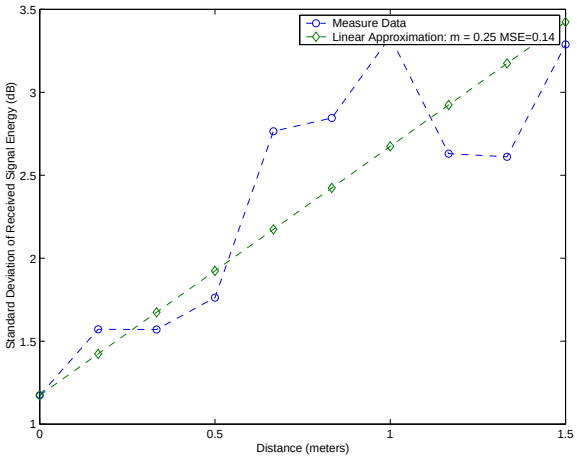

Figure 51: The standard deviation of received signal energy.

$$g(d) = 59.24 * exp(-\lambda * d) - 34.37 \quad (dB) \tag{35}$$

where d is the distance variable, and λ is 0.36/meters. The observation model given by (35) has a minimum mean square error of 5.58 dB^2 to the actual measured data.

The standard deviation of the received signal energy σ_{dB} is shown in Figure 51. It is almost linear and can be approximated by n($\cdot$), the measurement noise model, as

$$n(d) = 1.50 * d + 1.18 \quad (dB) \tag{36}$$

where d is the distance variable. The minimum mean square error of the approximation is 0.14 dB^2. The noise at 1 meter $n(1)$ is 2.68 dB, which is smaller than the value in [55] (3.92 dB). The reason for a lower noise is because the nodes in [55] are deployed in an entire office with many cubicles, desks, bookcases, etc. As a result, our setup has smaller interference. Nevertheless, this simple setup allows us to capture the observation and noise models with reasonable good results that are used by the PF component in Section 4.6.3.

4.6.2 Performance of the CE Component

Once the observation and measurement noise models are defined, they are applied to test the performance of the CE component in a 10 meters cube. Sensor nodes are deployed randomly in this cube with sensing range of 1.5 meters. The noise of the measurement is assumed to be Gaussian with standard deviation given by (36). Nodes A and B are separated by 1.5 meters. Node A is to determine the location of node B. One hundred simulations are ran to test the performance of the CE component. The performance of the CE component as a function of nodes is illustrated in Figure 52. As the number of nodes increases, the estimation accuracy increases as well. The results do match the expected performance of the CE component. Since more nodes are being deployed, the number of nodes in the set **S** increases allowing more nodes to help estimate the location of B from A. On the other hand, the performance of ME stays around the same level. The reason for this is because node B is lying at the edge of the sensing range of node A making it hard to determine the exact location. As shown in Figure 52, the estimation error with respect to the sensing range of 1.5 meters decreases from 45% with the ME to 15% with the CE component. As a result, the collaborative effort of the sensor nodes improves the estimation accuracy by 67%. Comparing to the example illustrated by Figure 44 that has a lowest CRB of 20%, the performance of the CE component in this experimental setup is 15%, giving a

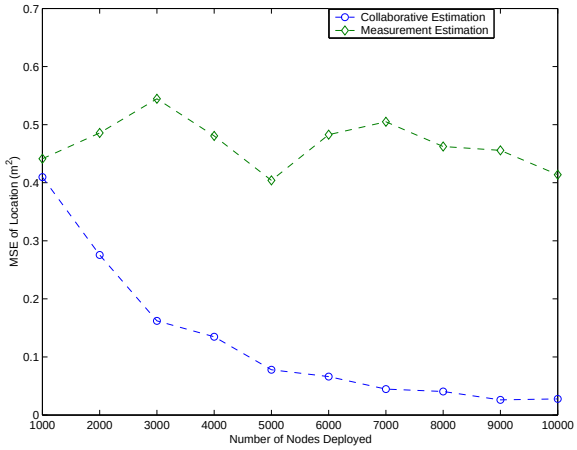

Figure 52: The performance of CE algorithm.

clue that it is approaching the CRB in its experimental environment.

4.6.3 Position Tracking

To track node B in 3-D space, three sensors are attached to node A. The observation and measurement noise models of these sensors are given by (35) and (36). Two movement models are used; one is a sine wave with frequency of 0.4 Hz, and the other is a random way point with random movement at speed in the range of ± 0.01 meters/second in the x, y, and z directions. The particle filter is modified from the code developed by [46]. The size of the particles ranges from 5 to 20, and 10 simulations are ran for each particle size. The MSE of tracking the sine wave is shown in Figure 53. As the size of particles increases, the MSE decreases, which is expected. The estimation error with respect to the sensing range of 1.5 meters decreases from 13% to 8% as N changes from 5 to 20. The posterior probability distribution is better approximated when the size of the particles is large. The average time took to perform the calculation for each particle size is depicted by Figure 54. The process time is linear with respect to the number of particles, which is expected since the computational complexity of the particle filter is $O(N)$.

The prediction error for a random way point movement model is illustrated in

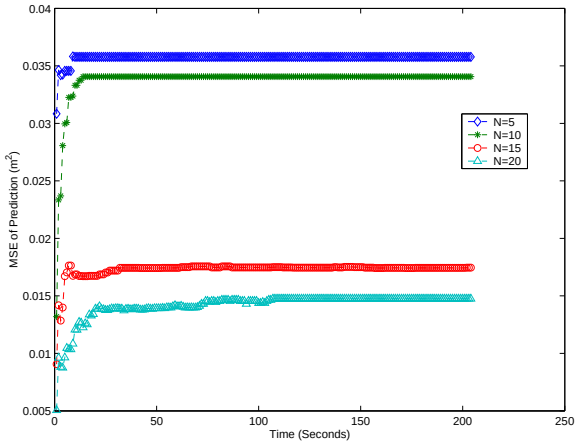

Figure 53: The MSE of predicting a sine wave.

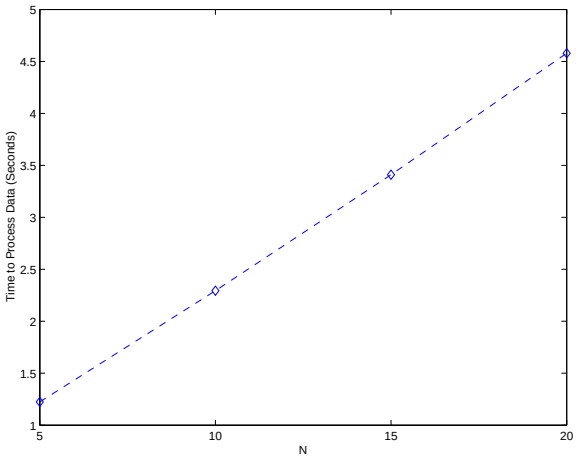

Figure 54: The process time as particle size increases.

Figure 55. As the particle size increases, the MSE is usually lower. The MSE for the random way point movement model is higher than the sine wave model. This is expected since random movement is harder to predict. For N equals to 10, the estimation error is 18% with respect to the sensing range of 1.5 meters. To show how the particle filter tracks the position of the neighbor node, the magnitudes of the estimated and actual positions of a random way point movement with respect to time are illustrated in Figure 56. The particle filter is able to track the general direction of the magnitude giving us a view of its performance.

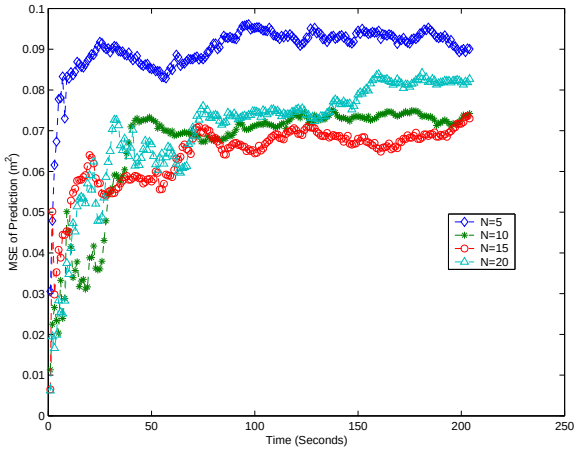

Figure 55: The MSE of predicting a random way point.

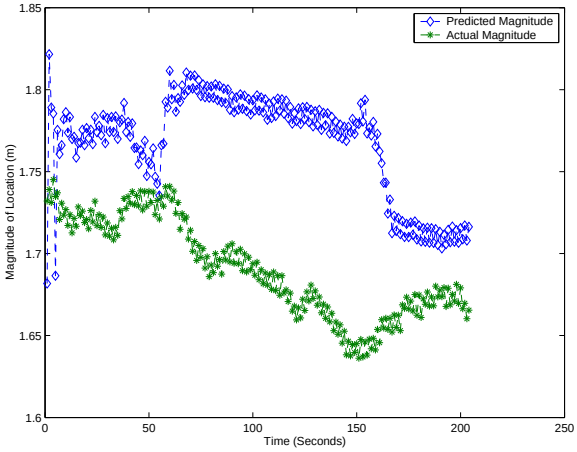

Figure 56: The magnitude of estimated position (random way point).

4.6.4 Localization Error

Since the PLF enables a node (e.g., node A) to determine and track the position of its neighbor nodes (e.g., node B), the error in locating a node in the sensor field is the sum of all the errors incurred per hop between the source and sink. The CE component contributes as low as 15% while the PF component adds another 8%. As a result, the error in locating a node in the sensor field may be as low as 23%. For example, if a sink sends a message to the source located at (100,100), the nodes that are within the radius of 23 meters ($\tau = 23\ meters$) centered at (100,100) may belief that the message is intended for them. Although this may sound ineffective, the cost

of deploying the PLF may only cost pennies. As stated in Section 4.3, the accuracy of the PLF is based on the following factors: (i) *the number of sensing units a node has*, (ii) *the quality of the sensing units*, (iii) *the density of the sensor nodes*, and (iv) *the number of samples used for PF*. As a result, the estimation accuracy may improve significantly if these factors are changed. In addition, the PLF is designed to be robust to sensor node failures, because it enables collaborative estimation and tracking that do not rely on any base station or beacon.

The PLF is an alternative technology used for localization since there are many applications of sensor networks that require different types of localization techniques. Although the location accuracy of PLF may be poorer when compared to a RSS technique with base stations, the PLF provides a cheaper and more robust infrastructure. For example, the base stations and sensor nodes use Datum ExacTime GPS and rubidium-based oscillators along with wideband direct-sequence spread-spectrum transmitters and receivers to perform localization. As a result, the location error with this setup is 2.30 meters in a 14 meters by 13 meters by 4 meters office [55]. According to the performance of the CE and PF components with cheap sensors and actuators, the PLF with RSS technique may provide an error of 10.58 meters since the longest length of the office is 46 meters and the estimation error of PLF is 23%. From the estimation error point-of-view, the PLF is not attractive, but the cost and robustness level of PLF is higher since the electronics are cheaper and the PLF does not depend on based stations for reference.

Chapter 5

Conclusions and Suggestions for Future Research

The research contributions of this thesis and suggestions for future research are described in Sections 5.1 and 5.2, respectively.

5.1 Research Contributions

This thesis presents three new schemes that may enable QoS applications in sensor networks. It consists of the following contributions for sensor networks:

1. QoS Routing Protocol
2. Time-Diffusion Synchronization Protocol
3. Distributed Perceptive Localization Framework

5.1.1 QoS Routing Protocol

In Chapter 2, the QSR protocol is described. The QSR protocol is verified by simulations to have better throughput than the DSR protocol when the node failure and packet loss probabilities are high, i.e., greater than 0.05. In addition, the QSR protocol embraces the increase of node density to increase the throughput of the streams. Also, the number of messages sent by using the QSR protocol may approach the DSR protocol by selecting different streams when the node failure and packet loss probabilities are low. Furthermore, the jitter and time required to reach the sink are near constant for different values of node failure and packet loss probabilities, and they are the lowest among the three protocols, i.e. Flooding, QSR, and DSR protocols.

When the sleep mode operation is turned ON, the number of messages that can be sent through the streams approaches the maximum, i.e., 180 messages for these

simulations. Using the stream that is designed to carry time sensitive messages, i.e., $S_1(1,0)$, the number of nodes participating in routing, the time required to reach the sink, and the jitter decrease as more nodes are deployed in the sensor field. In addition, the QSR protocol does not require each node to have an unique ID. As a result, only a small range of IDs is needed regardless if the number of sensor nodes is increased allowing random and progressive deployment of sensor nodes.

5.1.2 Time-Diffusion Synchronization Protocol

The constraint of requiring the nodes to maintain a similar time among the neighbors and throughout the network at conditions where outside timing sources, e.g., high power stations used to discipline the local time of the nodes in the network, may not be available due to distance and location, e.g., inside a cave or under water. With this constraint in mind, the time-diffusion synchronization protocol (TDP) is presented in Chapter 3. The TDP allows the nodes in the sensor field to reach an equilibrium time with a small tolerance from each other. Also, the TDP is analytically shown that it can be used to provide timing precision in the microsecond order. The precision is gated by the round trip delays among neighbor nodes. The convergence to the equilibrium time depends heavily on the rate of timing information message diffusion. This allows the designer to trade-off between convergence time and energy consumption. In addition, the TDP is thoroughly studied for both static and mobile sensor nodes. In both scenarios, the TDP enables the time in the network to converge to the targeted tolerance. Also, the time differences among neighbor nodes are small allowing smooth transition of time throughout the network. Besides enabling the time to converge and reach the targeted precision, the tasks for this process is distributed among the nodes in the sensor field. An additional advantage of the TDP is that it allows the designer to choose different τ values for different types of sensor networks depending on the purpose of the applications.

5.1.3 Distributed Perceptive Localization Framework

The PLF is described in Chapter 4, and it is used to detect and track the position of a neighboring node. The PLF can be extended to find any node in the sensor field provided that all nodes are connected. In addition, observation and measurement noise models of a low-end sensor and actuator are created. According to these models, the CE and PF performs well for both sine wave and random way point movement models. In addition, the accuracy of the position tracking can be increased by using a larger particle size N. Also, the PLF can be used to locate a node in the sensor field without using any fixed beacon for localization. In addition, the PLF is designed to be robust to sensor node failures, because it enables collaborative estimation and tracking that do not rely on any base station or beacon. Also, the CRB of the localization error is provided for both TOA and RSS techniques–providing a theoretical limit on the PLF localization accuracy.

5.2 Suggestions for Future Research

There are two research areas that should be look into: (1) routing between sensor nodes and actuators, and (2) sensor network management

5.2.1 Routing Between Sensor Nodes and Actuators

When actuators are deployed with the sensor nodes, a better and more efficient scheme may be needed to address the different types of traffics that exist among actuators and sensor nodes. Before resolving the routing issues, the topology of how sensor nodes and actuators are organized is important. The topology affects the traffic flow as well as creates congestion on critical routes. In the near future, the interaction between the actuators and sensor nodes may become critical. As of now, there is not much work with actuators.

5.2.2 Sensor Network Management

As illustrated in Figure 2, there are three management planes (power, mobility, and task) in the sensor network protocol stack. These planes can improve the effectiveness of the sensor networks, and more research should be done in these planes.

The power management plane manages how a sensor node uses its power. For example, the sensor node may turn off its receiver after receiving a message from one of its neighbors. This is to avoid getting duplicated messages. Also, when the power level of the sensor node is low, the sensor node broadcasts to its neighbors that it is low in power and can not participate in routing messages. The remaining power is reserved for sensing.

The mobility management plane detects and registers the movement of sensor nodes, so a route back to the user is always maintained, and the sensor nodes can keep track of who are their neighbor sensor nodes. By knowing who are the neighbor sensor nodes, the sensor nodes can balance their power and task usage.

The task management plane balances and schedules the sensing tasks given to a specific region. Not all sensor nodes in that region are required to perform the sensing task at the same time. As a result, some sensor nodes perform the task more than the others depending on their power level.

These three management planes are needed, so that sensor nodes can work together in a power efficient way, route data in a mobile sensor network, and share resources between sensor nodes. Without them, each sensor node will just work individually. From the whole sensor network standpoint, it is more efficient if sensor nodes can collaborate with each other, so the life-time of the sensor networks can be prolonged.

Appendix A

Derivation of Threshold Adjustment Value ε

In order to determine ε (equation (12)), the fraction of deployed sensor nodes ψ_i that can become a diffused leader node at Round (i) in Figure 21 needs to be determined. In addition, the energy μ consumed for each round of timing information message diffusion is required.

As a result, ζ is reduced by ϵ, i.e., ratio of μ over the maximum allowed energy level, after each round of timing information handshake. The derivation of ψ_i is as follows:

Round (1): $\psi_1 = \Gamma_{1,1}\rho$

Round (2): $\psi_2 = \Gamma_{2,1}\rho + \Gamma_{2,2}(\rho - \epsilon)$

Round (3): $\psi_3 = \Gamma_{3,1}\rho + \Gamma_{3,2}(\rho - \epsilon) + \Gamma_{3,3}(\rho - 2\epsilon)$

Round (4): $\psi_4 = \Gamma_{4,1}\rho + \Gamma_{4,2}(\rho - \epsilon) + \Gamma_{4,3}(\rho - 2\epsilon) +$

$\Gamma_{4,4}(\rho - 3\epsilon)$

.

.

.

Round (i): $\psi_i = \sum_{m=1}^{i} \Gamma_{i,m}(\rho - (m-1)\epsilon)$

At Round (1), the fraction of nodes that can become diffused leader nodes is set equal to $\Gamma_{1,1}\rho$, where $\Gamma_{1,1}$ is the coefficient; at Round (2), the fraction of sensor nodes

that did not participate as a diffused leader node has probability ρ of being reelected while the ones that participated has probability $(\rho - \epsilon)$, where $\Gamma_{2,1}$ and $\Gamma_{2,2}$ are the coefficients of ρ and $\rho - \epsilon$, respectively. The probability of being reelected is decreased by ϵ for the elected sensor nodes, because the randomly selected value λ is reduced by $(1 - \zeta)$ (equation (9)), where ζ is reduced by ϵ. By repeating the same evaluation at each round, a pattern emerges and gives the equation for ψ_i. The value $\Gamma_{i,m}$ is the m^{th} coefficient at Round (i). For example, $\Gamma_{4,3}$ is the 3^{rd} coefficient of Round (4), which is the coefficient of $(\rho - 2\epsilon)$.

The coefficient $\Gamma_{i,m}$ represents the fraction of nodes that has probability $(\rho - (m - 1)\epsilon)$, and it is derived as follows:

$$\Gamma_{i,1} = (1 - \rho)^{(i-1)}$$

$$\Gamma_{i,2} = \Gamma_{i-1,2}(1 - \rho) + \Gamma_{i-1,1}(\rho)$$

Note: $\Gamma_{i,2} = 0$ *for* $i < 2$

$$\Gamma_{i,3} = \Gamma_{i-1,3}(1 - \rho) + \Gamma_{i-1,2}(\rho)$$

Note: $\Gamma_{i,3} = 0$ *for* $i < 3$

.

.

.

$$\Gamma_{i,m} = \Gamma_{i-1,m}(1 - \rho) + \Gamma_{i-1,m-1}(\rho)$$

Note: $\Gamma_{i,m} = 0$ *for* $i < m;\ m > 1$

The coefficient $\Gamma_{i,1}$ is decreased by the fraction of nodes that is selected to be diffused leader nodes at every round. As a result, only $(1 - \rho)$ of the previous fraction of nodes will have probability ρ. The coefficient $\Gamma_{i,2}$ depends on the fraction of nodes

that has not been selected to be diffused leader node $\Gamma_{i-1,2}(1-\rho)$ and the fraction of nodes that has been selected to be diffused leader node $\Gamma_{i-1,1}(\rho)$. Basically, $\Gamma_{i,m}$ is composed of $\Gamma_{i-1,m}$ and $\Gamma_{i-1,m-1}$. This means that diffused leader nodes move from having probability $(\rho-(m-2)\epsilon)$ to probability $(\rho-(m-1)\epsilon)$ since their chances of being reelected as diffused leader nodes at the next round are decreased by ϵ. The general form for the m^{th} coefficient at Round (i) is given as $\Gamma_{i,m}$.

The $\Gamma_{i,m}$ coefficients can be further simplified in terms of ρ and i. They are given as follows:

$\Gamma_{i,1} = \Phi_{i,1}(1-\rho)^{(i-1)}$

$\Gamma_{i,2} = \Phi_{i,2}(\rho)(1-\rho)^{(i-2)}$

$\Gamma_{i,3} = \Phi_{i,3}(\rho)^2(1-\rho)^{(i-3)}$

.

.

.

$\Gamma_{i,m} = \Phi_{i,m}(\rho)^{m-1}(1-\rho)^{(i-m)}$

Note: $\Phi_{i,m} = 0 \;\; for \;\; i < m;\, m \;>\; 1$

where the coefficient $\Phi_{i,m}$ is calculated as

$$\Phi_{i,m} = \begin{cases} 1 & , \; for \; m = 1 \\ \sum_{j=1}^{i-1} 1 & , \; for \; m = 2 \\ (\sum_{v_{m-1}=1}^{i-(m-1)} (\sum_{v_{m-2}=1}^{i-(m-2)-v_{m-1}} (\ldots & , \; for \; m \geq 3 \\ (\sum_{v_1=1}^{i-1-\sum_{k=1}^{m-2} v_{m-k}} 1)\ldots))) & \end{cases} \tag{37}$$

with $i \geq m$ and $m-1$ levels of summation for $m \geq 3$, e.g., $(\sum(\sum \cdot))$ and $(\sum(\sum(\sum \cdot)))$ are 2 and 3 levels, respectively.

As a result, the value ψ_i and ε are calculated as

$$\psi_i = \sum_{m=1}^{i} \Phi_{i,m}(\rho)^{m-1}(1-\rho)^{(i-m)}(\rho-(m-1)\epsilon) \tag{38}$$

$$\begin{aligned} \varepsilon &= \rho - \psi_i \\ &= \rho - \sum_{m=1}^{i} \Phi_{i,m}\rho^{m-1}(1-\rho)^{(i-m)}(\rho-(m-1)\epsilon) \end{aligned} \tag{39}$$

where ρ is the fraction of sensor nodes that can become a master or diffused leader node; $\Phi_{i,m}$ is calculated by equation (37); i is the number of rounds within a τ period, which is approximated by $\lceil \tau/\delta \rceil - 1$; ϵ is the ratio of μ (equation (11)) over the maximum energy level.

Appendix B

Derivation of Cramér-Rao Bounds for TOA and RSS Techniques

The unknown parameter of interest is $d_{A,B}$, which is the distance between sensor nodes A and B. Since only one unknown parameter is of interest, the scalar CRB of the estimator variance is defined as

$$E\Big[(\hat{d}_{A,B} - d_{A,B})^2\Big] \geq -\frac{1}{E\Big[\frac{\partial^2 \ln(f_{X_{A,B}|d_{A,B}}(X_{A,B}|d_{A,B}))}{\partial d_{A,B}^2}\Big]} \tag{40}$$

where $\hat{d}_{A,B}$ is the approximation of $d_{A,B}$ and
$f_{X_{A,B}|d_{A,B}}(X_{A,B}|d_{A,B})$ is given by (21) with $X_{A,B} = T_{A,B}$ or (22) with $X_{A,B} = P_{A,B}$. The Fisher Information Matrix (FIM) is defined as $\mathrm{F} = -\mathrm{E}\Big[\frac{\partial^2 \ln f(\cdot)}{\partial d_{A,B}^2}\Big]$. Since only one scalar unknown is desired, the FIM is a 1 by 1 matrix. The log-likelihood of TOA density (21) and RSS density (22) are defined as

$$TOA : l_{T_{A,B}} = \ln\left(\frac{1}{\sqrt{2\pi\sigma_T^2}}\right) - \frac{(T_{A,B} - d_{A,B}/c)^2}{2\sigma_T^2}, \tag{41}$$

$$RSS : l_{P_{A,B}} = \ln\left(\frac{1}{\sqrt{2\pi\sigma_{dB}^2}}\right) - \frac{(P_{A,B} - \bar{P}_{A,B})^2}{2\sigma_{dB}^2}. \tag{42}$$

where $\bar{P}_{A,B}$ is given in (22).

The second partial derivative of (41) with respect to $d_{A,B}$ is determined as

$$\frac{\partial^2 l_{T_{A,B}}}{\partial d^2 A, B} = -\frac{1}{\sigma_T^2 c^2}, \tag{43}$$

and the second partial derivative of (42) is given by

$$\frac{\partial^2 l_{P_{A,B}}}{\partial d^2 A,B} = -\frac{10n}{d^2_{A,B}\sigma^2_{dB}\ln 10}\Big[\frac{10n}{\ln 10} - (P_{A,B} - \bar{P}_{A,B})\Big]. \tag{44}$$

Afterwards, the expected value of (43) is calculated and substituted into (40) to obtain the CRB of the error variance $\sigma^2_{d_{A,B}}$, which is bounded in the following way

$$TOA : \sigma^2_{d_{A,B}} \geq \sigma^2_T c^2. \tag{45}$$

Furthermore, the CRB of the error variance $\sigma^2_{d_{A,B}}$ for RSS is also obtained by (40) after the expected value of (44) is calculated. The error variance $\sigma^2_{d_{A,B}}$ is constrained as follows:

$$RSS : \sigma^2_{d_{A,B}} \geq \left(\frac{d_{A,B}\ \sigma_{dB}\ \ln 10}{10n}\right)^2. \tag{46}$$

References

[1] G. D. Abowd, and J. P. G. Sterbenz, "Final Report on the Inter-Agency Workshop on Research Issues for Smart Environments," *IEEE Personal Communications,* pp. 36-40, October 2000.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless Sensor Networks: A Survey," *Computer Networks (Elsevier) Journal,* pp. 393-422, March 2002.

[3] D. Allan, "Time and Frequency (Time-Domain) Characterization, Estimation, and Prediction of Precision Clocks and Oscillators," *IEEE Trans. on Ultrasonics, Ferroelectrics, and Frequency Control,* Vol. 34, No. 6, pp. 647-654, November 1987.

[4] D. Avitzour, "A Stochastic Simulation Bayesian Approach to Multitarget Tracking," *IEEE Proceedings on Radar, Sonar and Navigation*, Vol. 142, No. 2, pp. 41-44, 1995.

[5] P. Bauer, M. Sichitiu, R. Istepanian, and K. Premaratne, "The Mobile Patient: Wireless Distributed Sensor Networks for Patient Monitoring and Care," *Proceedings 2000 IEEE EMBS International Conference on Information Technology Applications in Biomedicine*, pp. 17-21, 2000.

[6] E. R. Beadle and P. M. Djuric, "A Fast Weighted Bayesian Bootstrap Filter for Nonlinear Model State Estimation," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 33, No.1, pp. 338-343, 1997.

[7] G. Bianchi, "Performance Analysis of the IEEE 802.11 Distributed Coordination Function," *IEEE Journal on Selected Areas in Communications,* vol. 18, no. 3, pp. 535-547, March 2000.

[8] P. Bonnet, J. Gehrke, and P. Seshadri, "Querying the Physical World," *IEEE Personal Communications,* pp. 10-15, October 2000.

[9] N. Bulusu, J. Heidemann, and D. Estrin, "GPS-less Low-cost Outdoor Localization for Very Small Devices," *IEEE Personal Communication,* Vo. 7, pp. 28-34, October 2000.

[10] N. Bulusu, D. Estrin, L. Girod, and J. Heidemann, "Scalable Coordination of Wireless Sensor Networks: Self-Configuring Localization Systems," *International Symposium on Communication Theory and Applications (ISCTA 2001),* Ambleside, UK, July 2001.

[11] BG Celler et al., "An Instrumentation System for the Remote Monitoring of Changes in Functional Health Status of the Elderly," *Int. Conf. IEEE-EMBS,* pp. 908-909, New York, 1994.

[12] A. Chandrakasan, R. Amirtharajah, S. Cho, and J. Goodman, G. Konduri, J. Kulik, W. Rabiner, and A. Wang, "Design Considerations for Distributed Microsensor Systems," *Proceedings of the IEEE 1999 Custom Integrated Circuits Conference,* pp. 279-286, San Diego, CA, May 1999.

[13] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks," *Proc. of the ACM MobiCom'01,* pp. 85-96, Rome, Italy, 2001.

[14] A. Cerpa, J. Elson, M. Hamilton, and J. Zhao, "Habitat Monitoring: Application Driver for Wireless Communications Technology," *ACM SIGCOMM'2000,* Costa Rica, April 2001.

[15] R. Colwell, "Testimony of Dr. Rita Colwell, Director, National Science Foundation," *Before the Basic Research Subcommitte, House Science Committe, Hearing on Remote Sensing as a Research and Management Tool,* September 1998.

[16] G. Coyle et al., "Home Telecare for the Elderly," *Journal of Telemedicine and Telecare,* Vol. 1, pp. 183-184, 1995.

[17] H. Cramér, *Mathematical Methods of Statistics,* Princeton University Press, Princeton, NJ 1946.

[18] B. P. Crow, I. Widjaja, J. G. Kim, and P. Sakai, "Investigation of the IEEE 802.11 Medium Access Control (MAC) Sublayer Functions," *IEEE INFOCOM'97,* pp. 126-133, Kobe, Japan, April 1997.

[19] L. Doherty, K. S. J. Pister, and L. E. Ghaoui, Convex Position Estimation in Wireless Sensor Networks, *IEEE INFOCOM01* , Anchorage, AK, April 2001.

[20] A. Doucet, N. de Freitas, and N. Gordon, *Sequential Monte Carlo Methods in Practice,* Springer-Verlag, 2001.

[21] J. Elson and D. Estrin, "Time Synchronization for wireless sensor networks," *In Proceedings of the 15th International Parallel and Distributed Processing Symposium (IPDPS-01),* IEEE Computer Society, April 2001.

[22] J. Elson, L. Girod, and D. Estrin, "Fine-Grained Network Time Synchronization using Reference Broadcasts," *To Appear in Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI 2002),* Boston, MA, December 2002.

[23] I. A. Essa, "Ubiquitous Sensing for Smart and Aware Environments," *IEEE Personal Communications,* pp. 47-49, October 2000.

[24] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next Century Challenges: Scalable Coordination in Sensor Networks," *ACM MobiCom'99,* pp.263-270, Washingtion, USA, 1999.

[25] S. Fischer, H. Koorapaty, E. Larsson, and A. Kangas, "System Performance Evaluation of Mobile Positioning Methods," *In Proc. IEEE Vehicular Technology Conference,* Houston, TX, USA, May 1999.

[26] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-Sync Protocol for Sensor Networks," *To Appear in ACM SenSys 2003,* Los Angeles, CA, November 2003.

[27] M. Gell-Mann, "What is Complexity?," *Complexity*, Vol. 1, No. 1, John Wiley and Sons, Inc., 1995.

[28] N. J. Gordon, D. J. Salmond, and A. F. M. Smith, "Novel Approach to Nonlinear/Nonlinear Gaussian Bayesian State Estimation," *IEEE Proceedings-F,* Vol. 140, No. 2, pp. 107-113, 1993.

[29] M. P. Hamilton, and M. Flaxman, "Scientific Data Visualization and Biological Diversity: New Tools for Spatializing Multimedia Observations of Species and Ecosystems," *Landscape and Urban Planning,* 21, pp. 285-297, 1992.

[30] M. P. Hamilton, "Hummercams, Robots, and the Virtual Reserve," *Directors Notebook,* http://www.jamesreserve.edu/news.html, February 6, 2000.

[31] S. Hedetniemi, S. Hedetniemi, and A. Liestman, "A Survey of Gossiping and Broadcasting in Communication Networks," *Networks,* 18, 1988.

[32] W. R. Heinzelman, J. Kulik, and H. Balakrishnan, "Adaptive Protocols for Information Dissemination in Wireless Sensor Networks," *Proc. of the ACM MobiCom'99,* pp. 174-185, Seattle, Washington, 1999.

[33] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-Efficient Communication Protocol for Wireless Microsensor Networks," *IEEE Proceedings of the Hawaii International Conference on System Sciences,* pp. 1-10, January 2000.

[34] C. Herring, and S. Kaplan, "Component-Based Software Systems for Smart Environments," *IEEE Personal Communications,* pp. 60-61, October 2000.

[35] IEEE 1588, "Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," 2002.

[36] T. Imielinski, and S. Goel, "DataSpace: Querying and Monitoring Deeply Networked Collections in Physical Space," *ACM International Workshop on Data Engineering for Wireless and Mobile Access MobiDE 1999*, pp. 44-51, Seattle, Washington, 1999.

[37] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," *Proc. of the ACM MobiCom'00,* pp. 56-57, Boston, MA, 2000.

[38] M. Isard and A. Blake, "Contour Tracking By Stochastic Propagation of Conditional Density," *European Conference on Computer Vision,* pp. 343-356, Cambridge, U.K., 1996.

[39] D. B. Johnson and D. A. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks," *Mobile Computing,* pp. 153-181, 1996.

[40] J. M. Kahn, R. H. Katz, and K. S. J. Pister, "Next Century Challenges: Mobile Networking for "Smart Dust"," *Proc. of the ACM MobiCom'99,* pp. 271-278, Seattle, Washington, 1999.

[41] T. H. Keitt, D. L. Urban, and B. T. Milne, "Detecting Critical Scales in Fragmented Landscapes," *Conservation Ecology,* [onliine]1(1):4, http://www.consecolo.org/vol1/iss1/art4, 1997.

[42] G. Kitagawa, "Monte Carlo Filter and Smoother for Non-Gaussian Nonlinear State Space Models," *Journal of Computational and Graphical Statistics,* Vol. 5, pp. 1-25, 1996.

[43] J. Levine, "Time Synchronization Over the Internet Using an Adaptive Frequency-Locked Loop," *IEEE Transaction on Ultrasonics, Ferroelectrics, and Frequency Control,* Vol. 46, No. 4, pp. 888-896, July 1999.

[44] J. Liu, D. M. Nicol, L. F. Perrone, and M. Liljenstam, "Towards High Performance Modeling of The 802.11 Wireless Protocol," *Proceedings of the 2001 Winter Simulation Conference (WSC 2001),* Arlington, VA, December 9-12, 2001.

[45] B. Mark and Z. Zaidi, "Robust Mobility Tracking For Cellular Networks," *In Proc. IEEE International Communications Conference,* New York, USA, 2002.

[46] R. van der Merwe, A. Doucet, J. F. G. de Freitas, and E. Wan, "The Unscented Particle Filter," *Advances in Neural Information Processing Systems (NIPS13),* December 2000.

[47] D. L. Mills, "Internet Time Synchronization: The Network Time Protocol," *In Z. Yang and T. A. Marsland, editors, Global States and Time in Distributed Systems,* IEEE Computer Society Press, 1994.

[48] D. L. Mills, "Adaptive Hybrid Clock Discipline Algorithm for the Network Time Protocol," *IEEE/ACM Trans. on Networking,* Vol. 6, No. 5, pp. 505-514, October 1998.

[49] R. Moses, D. Krishnamurthy, and R. Patterson, A Self-Localization Method for Wireless Sensor Networks, Eurasip Journal on Applied Signal Processing, No. 4, pp. 348-358, 2003.

[50] Y.H. Nam et al., "Developement of Remote Diagnosis System Integrating Digital Telemetry for Medicine," *International Conference IEEE-EMBS,* pp. 1170-73, Hong Kong, 1998.

[51] N. Noury, T. Herve, V. Rialle, G. Virone, E. Mercier, G. Morey, A. Moro, and T. Porcheron, "Monitoring Behavior in Home Using a Smart Fall Sensor," *IEEE-EMBS Special Topic Conference on Microtechnologies in Medicine and Biology,* pp. 607-610, October 2000.

[52] "Network Time Protocol (Version 3) Specification, Implementation, and Analysis," *Network Working Group,* Request for Comments: 1305, March 1992.

[53] M. Ogawa et al., "Fully Automated Biosignal Acquisition in Daily Routine Through 1 Month," *Int. Conf. IEEE-EMBS,* pp. 1947-1950, Hong Kong, 1998.

[54] T. Okutani, H. Watanabe, and T. Nisase, "Performance Evaluation of Multiplexing AAL2 Voice Traffic and TCP/IP Data at the ATM Cell Level," *IEEE ATM Workshop '99 Proceedings*, pp. 391-396, May 1999.

[55] N. Patwari, A. O. Hero III, M. Perkins, N. S. Correal, and R. J. O'Dea, "Relative Location Estimation in Wireless Sensor Networks," *IEEE Trans. on Signal Processing,* August 2003.

[56] C. Perkins, "Ad Hoc Networks," *Addison-Wesley*, 2000.

[57] D. W. Petr, R. R. Vatte, P. Sampath, and Y. Lu, "Efficiency of AAL2 for Voice Transport: Simulation Comparison with AAL1 and AAL5," *1999 IEEE ICC*, vol. 2, pp. 896-901, June 1999.

[58] E. M. Petriu, N. D. Georganas, D. C. Petriu, D. Makrakis, and V. Z. Groza, "Sensor-Based Information Appliances," *IEEE Instrumentation & Measurement Magazine,* pp. 31-35, December 2000.

[59] G.J. Pottie and W.J. Kaiser, "Wireless Integrated Network Sensors," *Communications of the ACM,* vol. 43, no. 5, pp. 551-8, May 2000.

[60] H. Saito, "Performance Evaluation and Dimensioning for AAL2 CLAD," in *IEEE Infocom '99*, pp. 153-160, vol. 1, March 1999.

[61] A. Savvides, C. Han, and M. Srivastava, "Dynamic Fine-Grained Localization in Ad-Hoc Networks of Sensors," *Proc. of the ACM MobiCom'01,* pp. 166-179, Rome, Italy, 2001.

[62] E. Shih, S. Cho, N. Ickes, R. Min, A. Sinha, A. Wang, A. Chandrakasan, "Physical Layer Driven Protocol and Algorithm Design for Energy-Efficient Wireless Sensor Networks," *Proceedings of ACM MobiCom'01,* pp. 272-286, Rome, Italy, July 2001.

[63] B. Sibbald, "Use Computerized Systems to Cut Adverse Drug Events: Report," *CMAJ: Canadian Medical Association Journal,* Vol. 164, Issue 13, p1878, 1/2p, 1c, 6/26/2001.

[64] S. Singh, M. Woo, and C. S. Raghavendra, "Power-Aware Routing in Mobile Ad Hoc Networks," *Proc. of the ACM MobiCom'98,* pp. 181-190, Dallas, Texas, 1998.

[65] K. Sohrabi, J. Gao, V. Ailawadhi, and G. J. Pottie, "Protocols for Self-Organization of a Wireless Sensor Network," *IEEE Personal Communications,* pp. 16-27, October 2000.

[66] M.A. Spirito and A. G. Mattioli, "On the hyperbolic positioning of GSM Mobile Stations," *In Proc. International Symposium on Signals, Systems, and Electronics,* September 1998.

[67] M.A. Spirito, "Further results on GSM mobile station location," *IEEE Electronics Letters,* 35(22), 1999.

[68] B. Subbiah, and S. Dixit, "ATM Adaptation Layer 2 (AAL2) for Low Bit Rate Speech and Data: Issues and Challenges," *1998 IEEE ATM Workshop Proceedings*, pp. 225-233, May 1998.

[69] W. Su and I. F. Akyildiz, "The Jitter Time Stamp Approach for Clock Recovery of Real-Time VBR Traffic," *IEEE/ACM Transactions on Networking,* Vol. 9, No. 6, pp.746-755, December 2001.

[70] W. Su and I. F. Akyildiz, "A Stream Enabled Routing (SER) Protocol for Sensor Networks," *Med-hoc-Net 2002*, Sardegna, Italy, September 2002.

[71] W. Su and I. F. Akyildiz, "Time Translation Algorithm," *Georgia Tech Report,* 2003.

[72] Y. C. Tay and K. C. Chua, "A Capacity Analysis for the IEEE 802.11 MAC Protocol," *ACM Wireless Networks Journal,* vol. 7, pp. 159-171, 2001.

[73] B. Walker, and W. Steffen, "An Overview of the Implications of Global Change of Natural and Managed Terrestrial Ecosystems," *Conservation Ecology,* [online]1(2), http://www.consecol.org/vol1/iss2/art2, 1997.

[74] http://www.alertsystems.org

[75] www.xbow.com

[76] Y. Xu, J. Heidemann, and D. Estrin, "Geography-informed energy conservation for ad hoc routing," *Proc. of the ACM MobiCom'01,* pp. 70-84, Rome, Italy, 2001.

Vita

Weilian Su was born in Guanzhou, Guandong, China on June 23, 1974. He received the B.S. degree in Electrical, Computer, and Systems Engineering (ECSE) from Rensselaer Polytechnic Institute (RPI) in 1997 with Summa Cum Laude. In addition, he received the M.S.E.C.E and Ph.D. degrees in Electrical and Computer Engineering from Georgia Institute of Technology in 2001 and April 2004, respectively.

Weilian received the United Federations of Teachers Scholarship (1993), Local 23-25 ILGWU Textbook Scholarship (1993), Governor's Scholarship (1993), and Xerox Scholarship (1994). He was a member of the National Dean's List (1994), Eta Kappa Nu (1995), and Tau Beta Pi (1995). He was a Research Assistant at RPI and received the Lockheed Martin Capstone Design Award in 1997 from the ECSE department of RPI.

Furthermore, Weilian was a Research Assistant at Jet Propulsion Laboratory in 1996. From 1997 to 1999, he was an ASIC Application Engineer, Communication Segment Microelectronics Division, at IBM.

In 1999, he joined the Broadband and Wireless Networking lab at Georgia Tech as a Research Assistant. In 2003, Weilian received the 2003 Best Tutorial Paper Award from IEEE Communication Society. In November 2003, he joined the Georgia Tech Research Institute helping to build early-entry combat networks, SIPRNET and NIPRNET.